

THE VIC
EXPLORING IT'S
INNER WORKINGS

SEPTEMBER 1981

COMMODORE CLUB NEWS

Volume 3 Issue 8

THE MICRO- MAINFRAME

What it's all about

PRINTERS

What the manuals won't tell you


BASIC AND MACHINE CODE

Linking the two - how it's done

EDUCATIONAL SUPPLEMENT

With the emphasis on VIC
programming



 **commodore**
COMPUTER

Contents

Editorial —introduction to the magazine.....	2
What the Papers Say —read what the rest of the world is saying about us.....	5
MMF 9000 —David Middleton gives us an in-depth report on its history and development.....	7
DOSSing around in BASIC —some known bugs and cures for them.....	11
Butterfield Speaks —continuing his new regular column.....	16
Day trip to Paris —Dave Middleton setting the controls for the sun with Martin Maynard of Audiogenic.....	17
Commodore Splits the Proton —a day out helping the B.B.C. with a PET and Chris Palmer.....	18
Reviews —Barry Miles looks at Command-0 from Skyles.....	20
More Reviews —This time 'Wordcraft' comes under the hammer.....	22
Basic Programming —continuing the collection of utilities with a number of readers' letters.....	25
Where to Start —Part 2 continuing our exploration into where to begin.....	27
Machine Code Programming —guide for beginners part 2, plus more tiny gems.....	28
Disk Use for Beginners —Part 5 of David Pococks beginners guide.....	32
Cassette Decks —some programs to help you out.....	35
Training Courses —on the road again.....	inside back cover

Educational Supplement

Editorial —What it's all about.....	1
Giving them away —Commodore U.S. donating PETs to schools.....	2
PET in Education Conference —Yes, yet another.....	3
Games and the PET —the role that games play in education.....	4
VICs —Paul Higginbottom explores the inner workings.....	5
Educational Workshops —the list keeps up to date.....	12

Next Month's Issue.

Next month we'll have all the regulars, the guides to beginners in machine code and the use of disk drives, Jim Butterfield talking about the new 12" 4032's, and the other features that have become a way of life in the magazine.

The major review next time will be covering the new challenger on the wordprocessing scene, namely Wordform from Landsoft, and we'll also be covering other products as well.

The programs in Basic and machine code will also be there, along with the special section on peripheral, and other programming features will include a detailed report on sorting, as well as plotting multiple functions on the printer.

A new guide for programming will be on writing your own compiler, and this will grow into a series of articles as time goes by, covering all aspects of this interesting task.

Finally, the special feature is back on communications again - more interesting developments with the PET in this increasingly important arena.

For the best PET software...

COMMAND-O.....	For Basic IV CBM/PET, 39 functions with improved "Toolkit" commands	£59.95 + Vat
DISK-O-PRO....	For Basic II PET, adds 25 commands including Basic IV, in one 4K rom	£59.95 + Vat
KRAM.....	For any 32K PET/CBM for retrieving disk data by KEYED Random Access	£86.95 + Vat
SPACEMAKER IV	For any PET/CBM, mounts 1-4 roms in one rom slot, switch selection	£29.95 + Vat
* USER I/O	For software selection of up to 8 roms, in any two Spacemaker Quads	£12.95 + Vat
PRONTO-PET....	Soft/hard reset for 40-column PETs	£9.99 + Vat
SUPERKRAM, REQUEST & KRAM PLUS will be available shortly		

We are sole UK Distributors for these products, which are available from your local CBM dealer, or direct from us by mail or telephone order. To order by cheque write to: Calco Software, FREEPOST, Kingston-upon-Thames, Surrey KT2 7HR (no stamp required). For same-day Access/Barclaycard service, telephone 01-546-7256. Official orders accepted from educational, government & local authority establishments

...at the best prices!

WORDPRO IV PLUS	RRP £395 less £98.75 = £296.25!
WORDPRO III PLUS	RRP £275 less £68.75 = £206.25!
WORDPRO II PLUS	RRP £125 less £31.25 = £93.75!
VISICALC	RRP £125 less £25.00 = £100.00!
TOOLKIT Basic IV	RRP £34 less £9.50 = £24.50!
TOOLKIT Basic II	RRP £29 less £7.25 = £21.75!

The items above are available by mail or telephone order at our Special Offer Price when purchased with any one of our software products. This offer is for a LIMITED PERIOD only. UK - ADD 15% VAT. OVERSEAS airmail postage - add £3.00 (Europe), £5.00 (outside Europe).

Calco Software

Lakeside House - Kingston Hill - Surrey - KT2 7QT Tel 01-546-7256

Editorial

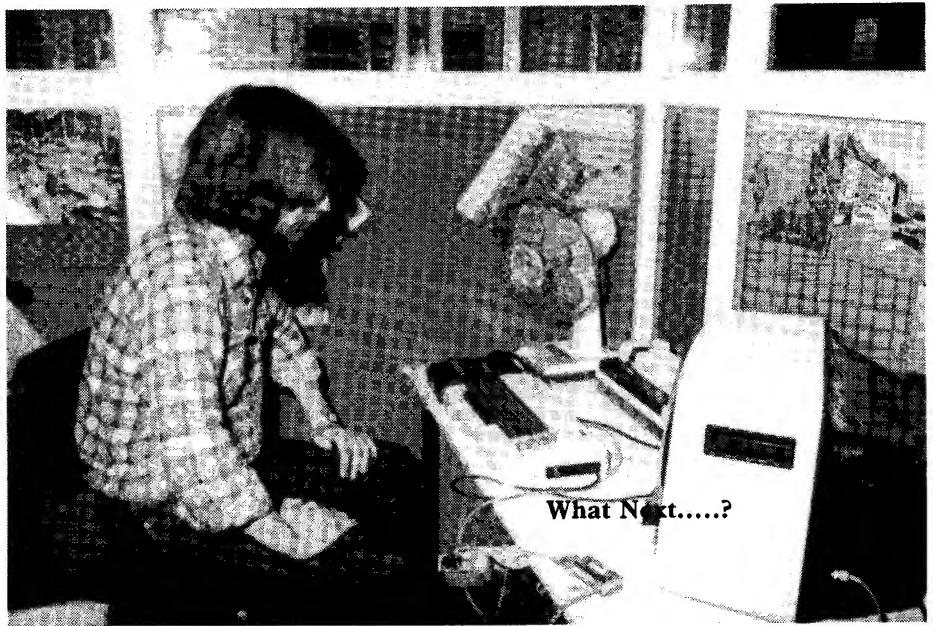
Here we are again, and it's quite hard to believe that this is the fifth newsletter I've done. It seems like about five minutes rather than five issues, but that must mean I'm enjoying the job I suppose. I only hope you're enjoying the results. It would seem so from the correspondence I've had so far, but as ever you're free to air your views via the usual address and telephone number at the end of the editorial.

I think you'll agree that we've seen an improvement in newsletters of late, and in size if nothing else. 52 pages a couple of newsletters ago, 60 pages last time (my printer's beginning to get fed up with me!), which can't be bad, and I'll be endeavouring to keep this up as the months go by. I can't guarantee you that number of pages every month, but I think we'll be up somewhere around there each time.

If there's an article on a particular subject that you'd like to see, write and let me know and I'll try my best to get such an article produced. It's your magazine after all, and also your interface to Commodore, which is why we regularly feature new products in it. Keeping you informed in other words.

One thing we have established is a regular base of articles that will be appearing each month. For instance, Jim Butterfield's monthly columns, the two beginner's guides on disk drives and machine code programming, the reviews of Approved Products, and in the last couple of issues the appearance of sections devoted to BASIC programming and machine-code programming and containing useful utilities for you to incorporate in your own programmes. The last two issues have also featured a section devoted to utilities for disk programming - this month we've moved on (back ?!) to cassette decks, and I'll try and keep on showing various peripherals as the magazines go by.

Appearing for the first time in this issue is a "what the papers say" column. I'm sure that you, like me, do NOT have the time to read every single newspaper throughout the country. Fortunately there are people employed to do this for us, and so any item of news that features Com-



modore duly gets reported and brought to our attention. For the benefit of those who don't read every column inch we've brought together a selection of the clippings throughout the month. Hope you find it interesting.

So, apart from the regulars, what else have we in store for you this issue? To carry on giving you as much information as possible, we have a report from David Middleton, of our technical department, on the MMF9000, the new micro-mainframe. He talks about where it will be used, and equally as important how it can be used, including a rundown on the languages available on the machine. Fascinating reading.

The review this time is of Wordcraft 80, one of the superb word-processing programs on the market at the moment. Very in-depth, and if you take this in comparison with the review last month of Wordpro, you should get a good idea of which system is the one for you.

One of the many things to emerge from the PET Show is the setting up of the Commodore Supply Co. for exporting British software and hardware overseas. I had a chat about this with the Sales and Marketing Manager for the company, namely the well-known Andrew Goltz, and the results can be read elsewhere in this newsletter.

Contrary to what appears to be a popular belief in some magazines,

Commodore have nothing against the B.B.C., and indeed we've helped them out on a couple of occasions. We editors seem to get involved in things like this, so inside we have a report on how myself and Chris Palmer of Adda Computers spent some time helping out in the first episode of a new B.B.C. science fiction serial. All good fun, and it's all inside.

As I was doing the B.B.C. episode, one of the other people here at Commodore, namely David Middleton, was jaunting about in the air from here to Paris, and having quite a time of it. What have David's doings got to do with a clean living magazine such as this? Well, it was to do with one of our Approved Products suppliers, and it was to get a preview of a number of new products, so to find out more I got Dave to produce an article for us.

Other items this time include a report on some of the Commodore printers, including a variety of facts and many valuable hints on programming them. To keep up my intention of giving you as much information on the variety of products that Commodore produce, there's an item on the new 12 inch screen 4032s, outlining some of the differences between them and the earlier 9 inch models in terms of programming the beasts.

Always ready to admit to mistakes, our technical boffins have put together a collection of known bugs

in both DOS 2.1 & 2.5 and BASIC 4.0, and also outlined solutions to those bugs. This collection is reproduced here for the benefit of those of you who are experiencing many hours of head scratching with not knowing what you're doing wrong! Hope it helps.

So this, plus all the usual goodies, is what we have in store for you this time around. I hope you enjoy it.

But what of the future? We will be definitely continuing as a monthly magazine and I'll be building up on the base of articles that will be appearing every issue - Jim Butterfield, the in-depth reviews, the beginners courses, the programming sections, and so on. But we can't get to the stage where you know precisely what's coming up each month - if that was the case you could all be editors! So, I'm forever on the look-out for new and interesting stories, programs etc. If you've got an application which you think may be of interest to others, or discovered some programming "oddity", or whatever, send your contributions in to the usual address. So you do your bit, and I'll continue to keep up my search!

One feature we'll certainly be keeping up is the educational supplement - this issue contains the third in the series so far, and there's plenty more where that came from. Nonetheless, educational users, this is your section, so as I said in the last paragraph if you've anything to contribute, you know the address! Let other people know what you're doing.

The "alternative" specials have so far concentrated on the Pet Show, in last month's issue communications, and in future months we'll be covering many varied aspects of C.B.M. equipment. There's so much being done these days that it's almost impossible to keep up, but we'll be doing our best, via not only the rest of the newsletter but these special sections as well. The communications one was well received (and I hope to be publishing another of those soon), and whenever we do another "special" in the future you can be assured that it will be with the same in-depth treatment.

So note the new address :-
Commodore Business Machines
675 Ajax Avenue
Trading Estate
Slough
Berks
Tel: Slough (STD 0753) 74111

And that is also the address to write to if you want to renew, or start, your subscription, but write to Margaret Gulliford, not me.

P.S. Thanks to those of you who wished me an enjoyable holiday - it was. We knew where the pubs were! I had a very good time, and to all the hospitable people in Wales who looked after us, thanks very much. So if you see some unusual pictures appearing in the newsletter, you'll know where they came from - some might say that they're all unusual!

More Training courses from Commodore

As well as Mike Gross-Niklaus starting up his training courses again, another source of training from Commodore, is a series of courses given by Spiro Omphalos. These are hardware courses on Commodore equipment, so if you want to know how they work, how to fix them when they go down, want to get the most up to date hardware orientated documentation, then the person to write to for all the details of where, when and how much, is :-

Fiona McCormick
Commodore Business Machines
813 Leigh Road
Trading Estate
Slough
Berks
Or, telephone Slough (STD 0753) 74111 extension 240 and speak to the lovely Fiona yourself.

Computer and Video Games

With the arrival of the VIC we'll be seeing an increase in the number of games programs that are on the market, from the highly professional plug in cartridge games to the beginners first attempts. And don't think I'm decrying beginners - everybody has to start somewhere!

Obviously the VIC isn't the only machine which has games written for it. There are a large number of other machines around (PET included) that have lots of games written for them. Games play an important role in computing, particularly in education where a game can go a long way to getting rid of a child's fear of the microcomputer. And after a hard days word processing there's nothing quite like blasting a few aliens to smithereens.

So, with this sort of existing and increasing market in mind, a new computer magazine is looming on the

horizon. Due to be launched in October is Computer and Video Games, from EEC Publications. If you've written anything that you'd like reviewed in the mag, feel free to send them (and me for that matter!) a copy of the program. Any kind of contribution will be welcome - your views on any aspect of writing computer games which you think fellow games fans would be interested in hearing about, for instance, would be gratefully received.

The names and address for further info are :-

Elspeth Joiner and Terry Pratt
30 - 31 Islington Green
London N1 8BJ Tel. 01 359 7481
I wish them all the best.

Mind Your Own Business

No, it's not me being rude, but it is the name of a magazine devoted to the businessman who wants or needs to know what is happening in the world of microcomputers. From September onwards they'll be having a special page devoted to the PET and its many uses in this area, so it should be well worth having a look at.

So, if you've got anything you'd like to contribute, or you'd like to see on the special PET page, give the editor a ring and have a chat with him. His name is Bill Gledhill, and his number is 01 771 3614. An amiable chap, and judging by the quality of the rest of the magazine it should be a welcome addition to the amount of information available on the PET.

Audiogenic

Enterprising Reading-based company Audiogenic, who recently took over the supply of Commodore's cassette based software, after being the manufacturers of same for some time, are on the look out for new software and hardware products to market.

After recently announcing a program that will run on 3032, 9" and 12" 4032's, as well as the 8032, (the 80 column Space Invaders!), and also Cosmiads and Cosmic Jailbreak for the 80 column PET you'll need to be good! But contact them, with details of what you've got, and who knows what might happen. Ring Reading (STD 0734) 595647, or alternatively write to :-

Audiogenic
34-36 Crown Street
Reading
Berks

Uarco Computer Supplies Catalogue Features Commodore Small Business Systems.

Uarco Incorporated, one of the largest manufacturers of business forms and systems, has chosen the Commodore CBMTM small business computer as the featured system in its new, forty-eight page computer supplies catalog.

Along with diskettes, disk packs, ribbons, binders and storage cabinets, the catalog carries a potpourri of software packages compatible with the Commodore system. Among the software programs available are basic book-keeping, inventory control, word processing and VisiCalc, the most popular business program on the market.

The Commodore CBM small business computer is the product of Pennsylvania-based Commodore Business Machines, Inc., one of the world's largest manufacturers of small business computers. The CBM comprises a CPU, a dual disk drive with 1 megabyte disk storage and a tractor printer. The equipment is sold under the Commodore name in the UARCO catalog.

According to Jack Resnick, UARCO's General Manager - Direct Mail Marketing, the company chose Commodore "because their small, desktop computer was by far, the most powerful and most reliable system on the market in a price range that is considered affordable by a small business."

"Unlike other small computers, Commodore's system has only three components and two cables. That makes it compact, easy to install, and easy to use", continued Resnick.

Although UARCO computers are able to buy the system directly from the catalog, the company has a large nationwide sales force that can arrange for support to a prospect. A toll-free hotline is also available for information and operator assistance.

As far as service is considered, UARCO has service centers throughout the country for on-site repairs for their Commodore customers. A loaner system is also available if, by chance, a customers system goes down.

According to Resnick, UARCO sees an enormous market potential for small computers. "Because of our diverse customer base and widespread

distribution, we are not only selling to the small business owner, but also to department heads in larger corporations. Our reputation as a service company of quality and integrity goes a long way when it comes to recommending the Commodore system", he added.

British Firms get Boost from Commodore

Europe's largest suppliers of microcomputers, Commodore Business Machines, announced today that it was setting up a new company - Commodore Supply Co - to export the best of British produced application software and peripheral hardware for its PET range of micros.

Bob Gleadow, managing director of Commodore's UK operations commented, - "The setting up of a separate export company reflects the tremendous success which British companies have enjoyed in developing products to operate together with Commodore microcomputers. Many of the products are already being sold, on an individual basis, in Europe and North America, and their joint marketing by our new export company will assist the independent companies which produce these products to achieve their maximum export potential".

The Company's initial catalogue will include a number of the latest "User Definable Application" programs such as OZZ and PETAID as well as the Mator hard disk drive, a high resolution graphics board developed for Commodore at Essex University, a word processing program and a BASIC compiler.

Sales and Marketing Manager of the new company Andrew Goltz, who as Marketing Support Manager, pioneered Commodore endorsement of third party products through the Commodore Approved Product

scheme. According to him Commodore have reversed the traditional rules of the mainframe computer industry by encouraging independent companies to develop products to plug into or work with Commodore processors even when a possibility existed that the proposed product directly competed with one marketed by Commodore. "Although this policy meant that, for example, a Commodore approved payroll program could be sold alongside one actually produced by Commodore, the deliberate competition so engendered, accelerated the development of products for the PET, and the number of mature applications available to the end user."

Certainly, Commodore's free market philosophy, has brought results. There are just under 100 products in Commodore's current "Approved Products" catalogue, and these formed the core of Commodore's recent "Pet Show" exhibition in London which attracted a staggering 11,500 visitors. "The international interest in British products was just amazing," said Andrew Goltz after the Show, "I really look forward to developing the export potential which exists for these products". With nearly a quarter of a million PET computers installed around the world, 50,000 of which are in the UK, the Commodore's export company will mean that UK manufacturers will be able to address a market place five times larger than the domestic market, so it looks as if Mr Goltz will have quite a busy time. He is nothing if not ambitious: - "I expect to have the framework of my initial European operation set up by November, after that we will have a crack at the United States."

For further information contact: Andrew Goltz (tel) SLOUGH (0753) 74111

Without Prejudice...

Sir,

With regard to the appearance of a picture in the last issue, it has come to our attention that our client, P.K. Coleman, is in some quarters being held to blame. Some days ago whilst imbibing medicinal draughts to reduce his understandable feelings of outrage, he became the subject of an itinerant photographer's interest. Information has been laid before us that the results of this unauthorised event may well be published.

What the Papers Say

This is the first time I've attempted a column such as this, so it was as much a surprise to me as I'm sure it will be to you, to find out just how diverse the references to PETs and their many uses are.

For instance, the Kentish Gazette reports on the setting up of a local Computer Users Group down in Canterbury, Kent. The co-founder of the group, Mr. John Wheeler, runs his business from home with the aid of a PET. Says Mr. Wheeler, "I use the computer to mastermind all my business affairs. It acts as a word processor, and I have now got it to such a pitch that I can manage a very busy organisation without a secretary." He goes on, "If you reckon the system cost £2,500, and assume a depreciation of as much as 25 per cent a year and compare that with the cost of even a part time secretary you will see just what an asset it is and what sav-

ings it makes." But it's not only the business that is helped ; his two children and wife are also reaping the benefits. "It is so versatile that the children are as much at home with it as I am". They are using some of the many good educational programs for their own use - "a sort of electronics shopping list" is just one example.

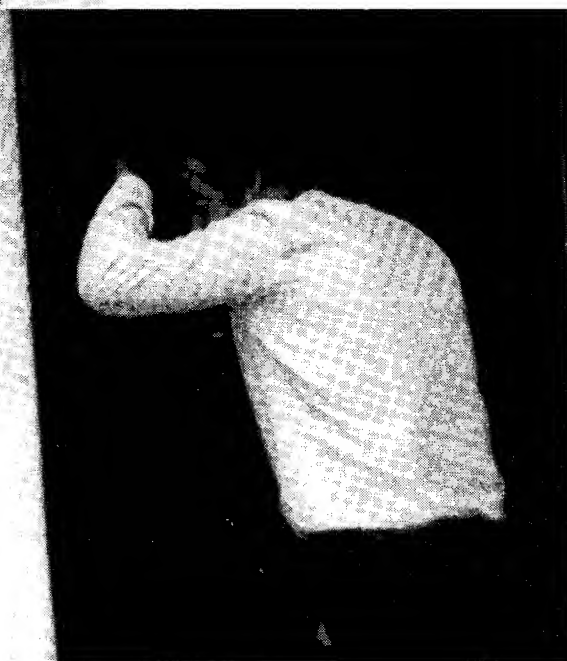
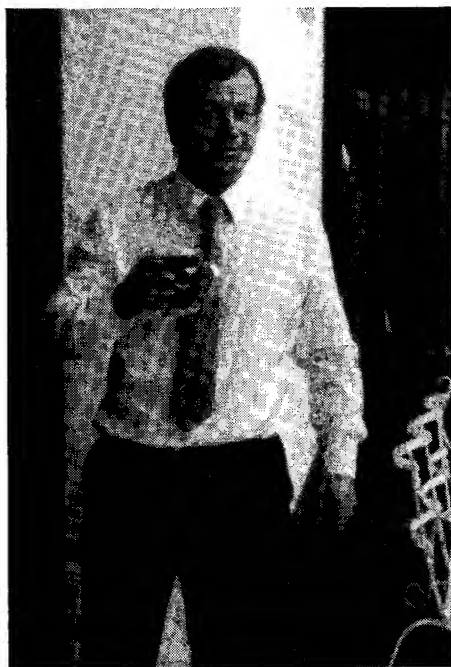
But the group doesn't just cover PETs. Anyone with an interest in computing is welcome to go along. They reckon to have already saved potential buyers who've gone along "for fun" considerable sums of money. For further details ring Richard Davis on Canterbury 68231 during normal office hours.

More groups springing up include one in York, according to Club Secretary. The Holgate WMC and United Services sowed the seeds for the York Computer Club, who held a press night recently. Chairman of the

club is Ken Thomas, an electrical engineer. "Information is swapped, and professionals in the field can also be called in for guidance. There are classes for beginners, and the advanced operator". They also cover programming in Basic and machine code, and have in addition a library of books, magazines and software. One thing that Club Secretary didn't mention is the address or 'phone number of the Computer Club, so if any of you up there are reading this, can you let me know and I'll get you in print.

Hair and Beauty is another mag, to feature a feature (knew I was running into trouble there!) on microcomputers in general, and had a number of things to say about computers and how they can help you. "They work easier, faster and more accurately than people and can also perform complex functions which are way beyond human capability. This, in ef-

Peter Coleman, of Heronview Limited, on a software evaluation day at Compssoft Limited!



Our client is a shy, retiring man of modest means and habits who devotes a great deal of his spare time to philanthropic pursuits and whose reward, it is rumoured, will be announced soon in the Margaret Thatcher Retirement Honours List. Consequently any slur on his character would be deemed by us to be a totally unprovoked attack which can only be redressed by a successful application for a considerable sum in damages.

Yours entirely without prejudice,

Messrs. Syntax, Error and Ready.

fect, makes computers very cost effective”.

Electronics Weekly was carrying information on calibrating of multimeters or any mass owned instrument for that matter. “This repetitive yet essential task can not be swept under the carpet, (although the betting is that many MDs would love to), because of MoD standards. One large percentage of an establishments instrument holding is multimeters, both analogue and digital. These need a combination of AC & DC volts, AC & DC current and resistance stimuli, standardised to accuracies four and ten times greater than the units under test (UUT)”

Well, the answer for handling all these parameters has arrived from PPM in Woking, using a system based around a Commodore PET and disk drive. “Amortising the cost of the total system over say ten years, would represent a significant saving in time : 20 minutes per UUT instead of an hour or so” said Electronics Weekly. For more information ring PPM on 04867 80111.

A couple of other electronics mags. were giving us mentions recently. The Electronics Times had two things to say. One was about the successful linking of a bacterial counter to a micro. by Aughton Microsystems. “The counter uses a laser beam to analyse milk samples, and the Commodore PET 4032 initiates the counting procedure”. Aughton are on 051-548 6060. The other was about a “low cost 80 word vocabulary printed circuit voice recognition board with an accuracy of 99%”. Produced by Auricle, of Cupertino in California, the board uses the services of a host computer, like, for instance a PET

Staying in the medical arena, Medical Laboratory World was running a feature on an “Image Analyser converted to colony counter”. They had this to say : “The addition of a specially developed lighting system for petri dishes to Micro Measurements Systems III small image analyser has converted the instrument into a sensitive microbiological colony counter. It can be interfaced via an IEEE interface to a range of low cost computers e.g. Commodore PETs.” They later went on “The System can be used in the microbiology laboratory for assessing the area of zones of inhibition etc. A further refinement....makes it possi-

ble to count agar plates. It can also detect and count malarial trophozoites in thin blood smears”. Fascinating stuff.

The third electronics journal is Electronics Engineering, who were mentioning a “shaft position encoder with PET interface from Cetronic”. I won't go into the technical details (mainly because I can't understand them!). but if you're interested Cetronic are on 0920 871077.

The Plastics and Rubber Weekly (I knew we'd get onto my kind of magazine in the end!) had a brief item on the “Manchester Section of the Plastics and Rubber Institute presenting the Manchester polytechnic Faculty of Science and Technology with a PET for use in the teaching of students on polymer courses”.

Systems International also had a mention for Aughton Microsystems, saying ‘Aughton's range of cards includes 8 and 12 bit a-d convertors, t.t.l. compatible digital i/o card, relay driver cards and thermo-couple and resistance bulb temperature input cards. All i/o cards can be operation-conditioned, i.e. mV-V and mA-A for analogue, or voltage free for digital’. Obviously busy chaps at Aughton.

What else have we been up to lately ? Well, in Office Equipment we read about “Termipet from MSI Data International” (contact Catlands, on 0625 527166, the distributors for further info.), and this is what they went on to say : “consists of an MSI 77 hand-held data capture terminal and interface which is used to transfer data into the memory of the PET microcomputer using keyboard entry or wand scanning of bar codes remote from the installation. It is claimed that all data can be transferred to the PET in under three minutes”. Works on 3000, 4000 and 8000 series machines

Lots of medical stuff happening. Computing carried a story on “Micro firms race to grab high street deal”, and concerned itself with “the race to capture a large slice of the high street chemist trade”. A big package just launched is the Monarch, from Sussex-based Channel Business, which “runs under Commodore's 8032 and features a cash register that can record receipts into stock and other transactions as well as computer programs to handle processing. A spokesman for Channel said ‘Monarch was developed for phar-

macists by pharmacists so they need no previous knowledge of computers”.

That august journal The Engineer had a long item on “Is Japan losing machine tool glamour ?”. It seems that a highly successful importer of Japanese machine tools, Mills Marketing Services, is moving away from Japan into other areas, claiming that Japan had lost some of its competitive edge. Mills current biggest attraction is the Nakamura Tome lathe, fully equipped for unattended operation. It has a whole host of features, and all but two of the automation equipment items are British. And where do we come in ? “Work coming off the lathe is transferred by the robot - either every time or on a sampling basis - to the automatic gauging station which can measure up to 32 dimensions and compare them in its Commodore PET computer with the nominal dimensions and the tolerances on each”. We get everywhere!

It seems that PETs are making rapid in-roads into industry lately. Roofing Cladding and Insulation Journal carried an article on Wind Loadings, as the Revised Code of Practice for Slating and Tiling has been revised again this year. To enable the Code requirements to be applied to all their products, Redland Roof Tiles has published a new technical guide entitled “Wind Loadings”, which incorporates a special rotary calculator. In developing the rotary calculator Redland also developed a micro computer program, anticipating the trend towards the use of desktop micros in the building and roofing industries, and is now offering the program as suitable for the Commodore PET. “A complete wind loading and fixing specification can be obtained accurately in under one minute with a print-out if required”.

We also made it into the Foundry Trade Journal, where they were describing a series of programs developed by the BNF Metals Technology Center, on the estimation of the weight and cost of castings, which are designed to run on 32K machines. These programs are equally applicable to all types of foundry, and use standard data available in any foundry to do the estimating tasks quickly and accurately. Further details are available on 02357 2992.

The Micro Mainframe

Introduction

The system can be seen as two independent machines:—

1. A standard 8032 with 64k of bank switched RAM.
2. A 6809 machine with 96k of RAM and 18k Operating System.

1. As an 8032

This would mean that the MMF9000 can be bought to run any current software available for the PET, eg. VisiCalc. It also means that software can be written to take the extra 64k of RAM into account. The MMF9000 has the advantage over the 8096 in that software can still be written in Microsoft BASIC and make use of the extra RAM. How it would make use of the RAM is difficult to say as the RAM resides at \$9000-\$9fff; ie. just after the screen. Obviously the RAM could be used to hold machine code routines which are at present in the lower 32k of RAM. Data could be stored in the range but BASIC is too slow to make it practical and there is little value in writing a whole new program in machine code as the 6809 processor is very much more powerful.

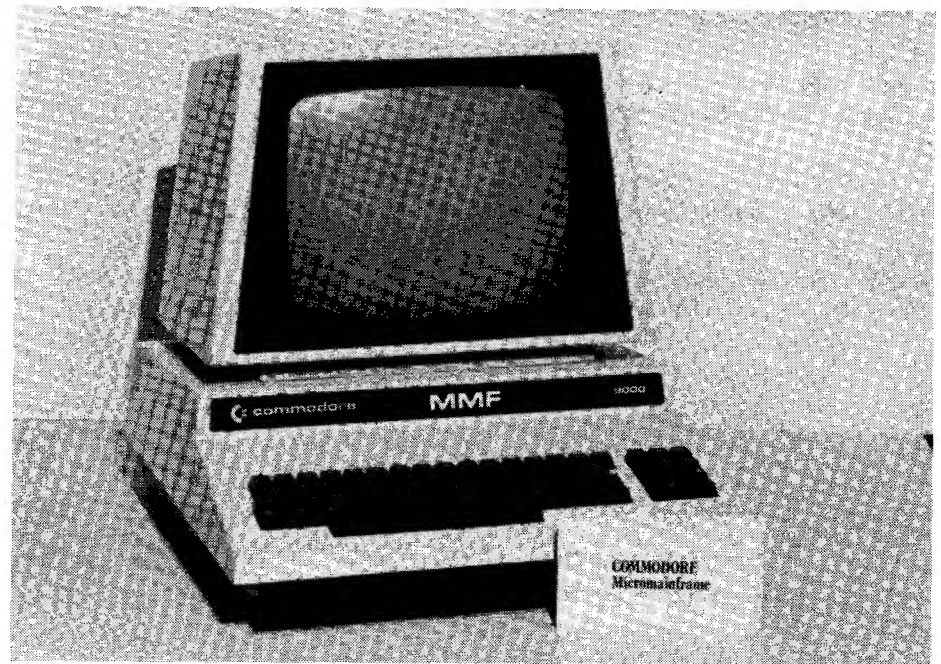
The only people who will be writing programs to make use of the 64k of RAM on the 6502 side of the machine are those with large machine code programs. Eg VisiCalc, WordPro and WordCraft. New applications will make use of the MMF9000.

One interesting point is the fact that many manufacturers use security or add-in chips which work in the \$9000 slot. These can now be down loaded from disk and stored in RAM. Careful use of the bank switching would allow the user access upto 16 different chips at once.

In conclusion, the 64K of RAM offers few advantages to the 8032 user but the system allows access to be maintained with one of the best supported micros on the market in terms of software and hardware add on.

2. The MicroMainframe 9000

The MMF9000 is a powerful computer capable of running advanced software. The machine uses a second generation 6809 microprocessor



which has 16 bit data registers and an 8 bit data bus. The 8 bit bus means that only 64k of RAM can be addressed directly, the same as the 6502, but there is a significant advantage in being able to do 16 bit arithmetic. Another advantage of the 6809 is the fact that it has a symmetric instruction set. This means that interpreter/compiler become much easier to write as there are no special cases which have to be taken into consideration.

It is interesting to trace the development of the MMF9000 as this has a great deal to do with how the final system operates.

Waterloo University were looking for a personal computer which could run powerful, portable software but virtually nothing was available. They had been using 8032's for some time and were developing software to run with the 6502. There are usually two free sockets on an 8032 and these were soon filled with their own code.

The next stage of development came when it was decided that the region \$9000-9FFF should be bank switched EPROM and the \$A000 slot would control which of the banks were operating.

The final change was fairly large. Using EPROMs for a development system caused considerable delay in debugging and bank switched RAM would be far more appropriate. Also there were problems with the 6502 as

a processor and so the necessary modifications were made to change the 6502 for the 6809.

It was now possible to use the redundant ROMs from \$B000-\$FFFF for code to run with the 6809. This is called the Supervisor and contains a monitor, an operating system for controlling I/O from the screen, keyboard, RS232, IEEE. It also contains routines for performing maths, handling the Commodore disk system and other general purpose routines which many different programs would find useful.

The objectives of the MMF9000

Waterloo University have one of the largest Computer Science courses in the world and even the large IBM they have on campus is unable to keep up with the demand. By using personal computers networked onto the IBM it is fairly simple to reduce the workload put onto the large computer, thus releasing it to do the job it is very good at, high speed number crunching.

In a teaching environment, most of the jobs run are very small with many errors. The MMF9000 has been designed to counter this by bringing into play powerful interpreters capable of running reasonable sized jobs on a stand alone computer. Another feature of the MMF9000 is that all of the interpreters have been

produced by a language called 'Whistle', this system produces code which is portable between machines with the claim that a system can be implemented very quickly on another machine. Thus the interpreters running programs on the MMF9000 can be easily moved to the largest main-frame computer thus enabling the program to be run hundreds of times faster.

In a testing environment, interpreters are far more useful than compilers as it is very much easier to debug programs. One drawback is that code does not run as fast as compiled programs. This is not a drawback to students, who are after results, not speed.

Obviously the MMF9000 has not been designed for commercial applications but at the same time many software authors will purchase the machine to write programs which will be used commercially, this will be because of the powerful languages which have been implemented on the MMF9000. However, until compilers become available which will put code into the bank switched RAM the machine will not be taken seriously by business software authors. When the compilers become available the MMF9000 will be able to run programs unthought of on a Commodore computer.

The languages available on the MMF9000.

There are four languages on the MMF9000 at present, these being:-

1. BASIC
2. FORTRAN
3. PASCAL
4. APL

There is also a very powerful assembler and COBOL is under development.

All the languages are interpreted.

1. BASIC.

This language bears almost no relationship to Microsoft BASIC. It is very highly structured which makes programs easier to read, write and debug. It also recognises names up to 31 characters in length and can call procedures by name. Other advantages are, the ability to continue running a program after modification has been made to a line, greatly speeding up debugging. In Microsoft BASIC a function can only be on one line, in WatBASIC the function can be as large as necessary. Input/output to

disk has been simplified but the language can handle Sequential and Relative files with ease. The interpreter is in the region of 32k of code which gives some indication of its comprehensive capabilities compared to only 8k for Microsoft. The interpreter uses an unusual way of tokenising programs which maintains the large system outlook while giving significant speed improvements against normal interpreters. The interpreter loads programs in source format and tokenises everything as it comes in from disk. This leads to significant speed increases when the program is run. Waterloo claim that the structured nature of the language leads to faster execution of large programs because there is very little searching involved, unlike the more normal GOTO and GOSUB commands of Microsoft BASIC. On small programs the language runs much more slowly.

2. FORTRAN.

The FORTRAN implemented by Waterloo is unlike ANSI standard FORTRAN-77 in so many ways that it is more like BASIC with FORTRAN extensions. In many respects the language is far better than FORTRAN-77 but it is not going to please many users when they find that it can not completely access the massive technical library of programs available. The main value of FORTRAN is its ability to pass variable data between routines allowing massive libraries of useful routines to be held on disk. Most of the commands associated with this ability are missing from WatFOR so the use of the language in connection with existing programs such as the graphics package GINO are missing. Being an old language FORTRAN is designed around card files, this is the biggest drawback with using FORTRAN and puts many people off. However, WatFOR has removed the need for taking care of column positions and introduced many new facilities which are not available in FORTRAN-77, especially in the area of structured programming. This means that many users who learn the language are going to get a very big shock when they try to implement programs using standard FORTRAN on a different machine. FORTRAN was designed to be compiled but compilers are not very useful in an education environment where debugging is of great im-

portance, WatFOR is therefore interpreted and will run fairly slowly.

3. PASCAL.

The Pascal interpreter follows the standards laid down by the ISO and contains a superset of commands of Pascal as laid down in the report by Wirth. The language is not the same as UCSD Pascal which has gained wide acclaim with users around the world. The big limitations of standard Pascal still remain, ie. there is poor file and string handling. In all other respects, however, the language is faithful to Wirth. Pascal is interpreted directly and does not pass through a p-code interpreter, thus it is slower than other implementations.

4. APL.

APL (A Programming Language) occupies the full 64k of bank switched RAM and is a complete implementation of the language. IBM implemented APL on its machines but it was never in much favour with IBM who preferred PL/1 as their brain-child. The language has many followers and there are a number of large computer companies which offer time-shared APL services, at a cost. APL is considered by many to be a 'computer freaks' language and indeed it would seem that it is ideal for the 'freak' because of its complex operators but many large companies (British Leyland for instance) use APL. This is because the language can define data matrices very easily, and once the operators have been mastered it is simple to perform 'What if..' calculations very quickly, to a much more advanced state than VisiCalc.

6809 Assembler

The Assembler development is a very powerful tool for writing code. There are a number of significant advantages in using the 6809 as opposed to the 6502 for new development work. The 16 bit registers are the main advantage but the instruction set is far more comprehensive eg. branch to subroutine, branch always, transfer between any register, 8 bit multiply. There are also two stacks, a hardware stack and a user stack. The user stack is vitally important to the operation of the bank switching.

The assembler has, like all other Waterloo software, been given structured programming commands such as IF...ELSE...ENDIF and the pro-

grammer does not have to worry about branch instructions, these are all handled by the assembler.

The bank switched RAM can be used for large machine code programs quite simply by specifying the banks to be used by the Linker, all the control software is then set up as the code is linked together. Thus to the programmer the 64k of bank switched RAM can be looked upon as continuous RAM. Some care is necessary in setting up the bank switched RAM. For instance it is necessary to ensure that all routines are shorter than 4096 bytes in length (Error messages are generated if a bank is overfilled). It is also necessary, if speed is of importance, to ensure that routines which are called a lot are grouped together in the same bank of RAM thus removing the need for continual switching.

Bank Switching

The bank switched RAM is available to both processors. The banks are selected by putting a number between 0 and 15 into a bank switch register, hardware will then perform a bank switch. So it is possible to bank switch the RAM from BASIC by a POKE61424,n and the extra RAM can then be used for storing data. The problem with using BASIC however, is that the RAM can only be accessed by PEEK and POKE. It is necessary to delve into machine code if the RAM is to be used effectively.

The 6809 can access the bank switched RAM in the same manner as the 6502. ie code in the lower 32k of RAM accesses data in the 64k of bank switched RAM. It is only sensible to use this method for large programs. Instead of putting the program into

normal RAM, it resides in the bank switched RAM and performs bank switching internally. There are some problems involved in performing an internal bank switch, for instance, it is not possible to jump to a subroutine in a different bank without first switching banks and it is not possible to switch banks before jumping to a subroutine. The solution to this seemingly impossible situation is to handle the bank switching external to the banks by using a 16 byte subroutine, this keeps track of the calling bank and the return address by pushing data onto the user stack. See figure 1 (this page) and figure 2 (overleaf).

How can MMF9000 be used? In Education.....

There is an increasing need for students in all disciplines of education to use computers. This puts a rising burden onto the mainframe machines used for time sharing between students. However, only a few students require the power these machines offer, the rest are usually wrangling with the intricacies of bugs in relatively small programs.

It is important that a computer be easy to use, the days when the only interactive part of debugging code was passing a card deck over to an operator are, hopefully, finished. Interactive debugging is very time consuming but to a personal computer this is fairly irrelevant, a one to one relationship with the user usually gives instant response to most debugging commands and users normally only run small programs.

The aim of the MMF9000 is to reduce the need for students to work on the mainframe computer directly,

leaving it to do the problems it is good at, very fast number crunching. The idea is to have a number of MMF9000 networked onto the mainframe allowing the user to download programs from the mainframes disk store into RAM. The program can then be run, altered or debugged at will.

The question now must be, why use the MMF9000 in the first place. The MMF9000 has four languages implemented on it so far, BASIC, FORTRAN, PASCAL and APL. These are all high level languages, implementing as far as possible the idea of structured programming. The languages are all large (BASIC is 32k APL fills the full 64k of bank switched RAM) and very comprehensive in their abilities.

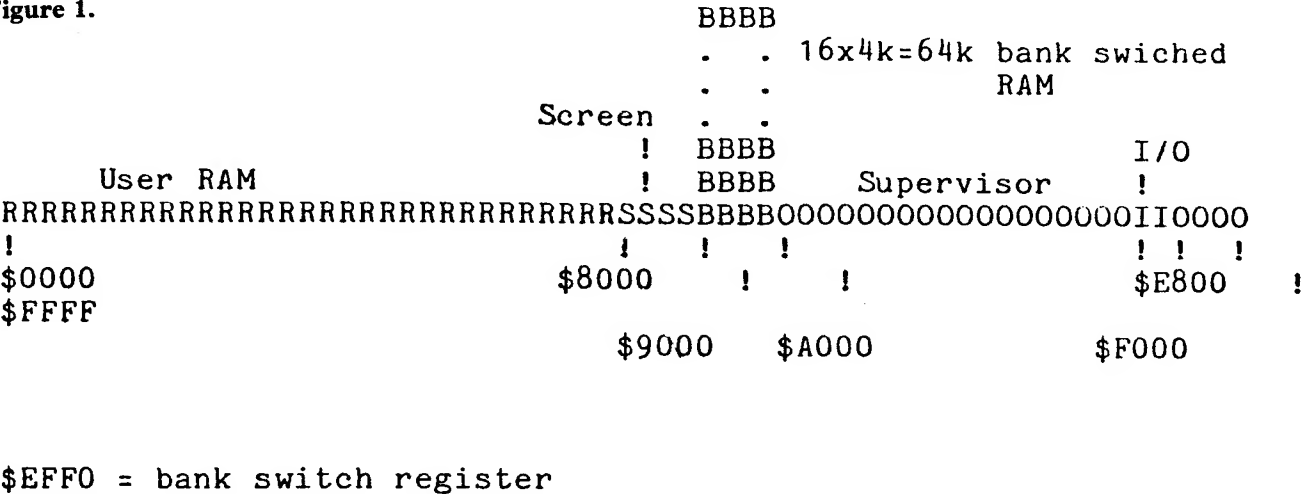
All the languages have been developed using 'Whistle', this is a high level language in its own right but with the ability to generate code which will run on any computer. Thus programs developed on the MMF9000 will also run on the largest IBM, assuming that the language interpreter has been implemented on the IBM. The portability of the languages assumes a generalised operating system which has to be implemented on the mainframe, this would be the most complex part of getting the language interpreters to run.

Thus there is now a system whereby a user can develop programs on the MMF9000 and when the program becomes too large, or needs to execute faster the program can be run on a larger machine.

Commercially....

For 95% of the commercial users the MMF9000 will be used with the

Figure 1.



Commodore disk systems and a printer in a stand alone environment. The cost of linking a micro to a main-frame is prohibitive to most small business users.

The MMF9000 offers the commercial user a number of advantages against a standard 8032 or other micro-computers. Despite the claims for PASCAL, it is not a commercial language, it lacks too many facilities, especially for peripheral control and file handling. FORTRAN is a scientific language. This leaves COBOL and BASIC. A COBOL interpreter is under development now and should be available by the start of 1982. COBOL is a language which is very much out of date but still retains wide usage because of the number of people who were bought up on it.

BASIC is not the ultimate language

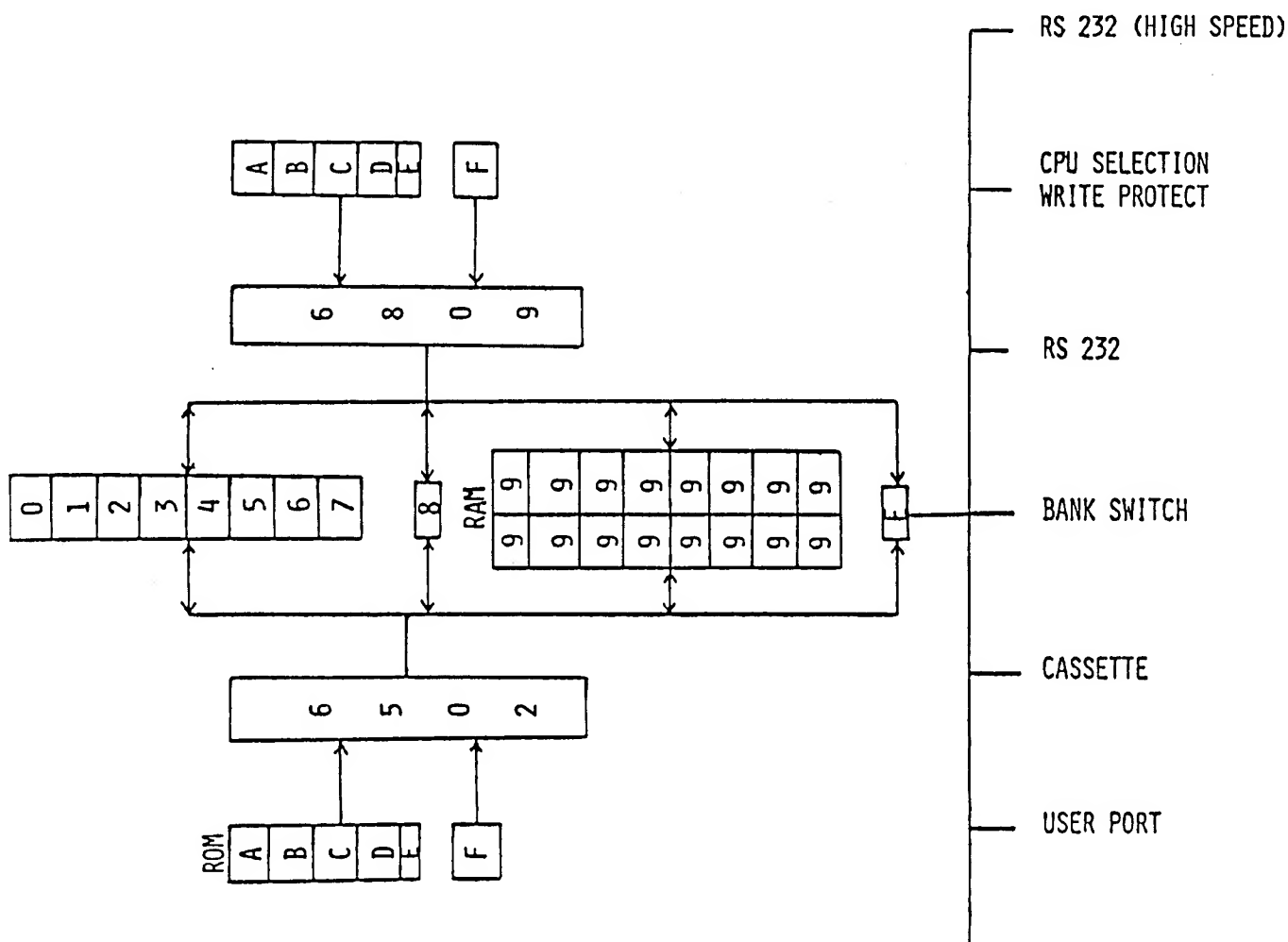
and never will be but unlike any other language it has no real standard thus people develop it to suit their needs. Waterloo BASIC is highly structured, with long variable and procedure names up to 31 characters. The language does not resemble Microsoft BASIC in any form so there would have to be a learning period for programmers to convert to the new language. However, once this has been accomplished the speed with which code can be produced will be dramatically faster than at present.

There are no compilers for any of the languages at present, BASIC and APL are usually interpreted only. PASCAL is usually compiled into p-code which runs faster but not considerably so. FORTRAN and COBOL were designed as compilers so the versions produced by Waterloo

tend to run slowly. All of the languages at present are used as development tools and as such are very powerful. For use in a commercial environment it is becoming increasingly more necessary for programs to run quickly, to achieve this programs have to be compiled. Microsoft BASIC as used on the standard CBM series machines have a number of BASIC compilers available. The language was not designed to be compiled and so there is very little increase in speed. However the highly structured nature of the Waterloo BASIC lends itself very well to compilation and so will outstrip Microsoft in this respect.

Edited by :—
Dave Middleton
Software Technical Support.

Figure 2



Dossing Around with Basic

DOSSing around with BASIC

In response to recent telephone enquiries from dealers about the bugs in DOS 2.1, DOS 2.5, BASIC 4 and controlling the 8026 and 8027 printers, we have documented the problems and possible solutions in this report.

DOS 2.1 and 2.5 Bugs

The following bugs are to be found in both DOS 2.1 and 2.5:—

1. **Disk Full Message Too Late**
The disk operating system is supposed to check the number of blocks still free on the disk and when only two remain, the 'Disk Full' error message is supposed to be sent and the file closed thus retaining some of the information.

What actually occurs is that the 'Disk Full' message is sent when there are no free blocks on the disk. The file can not then be closed properly. The BAM is updated to show that there are no free blocks and the file shows the number of blocks written to disk as zero. The file is also tagged as being unclosed by the use of an asterisk. A 'COLLECT' command has to be used to clear the files from disk and recover the lost blocks.

Using a save with replace causes the disk to write the new information to an unused area of the disk and then when this is successful close the file and scratch the old version. When the 'Disk Full' message error is sent, output terminates and the new file does not show on the disk. The BAM is updated to show zero blocks free. The old file which was to be replaced remains untouched. A COLLECT is needed to recover the used blocks by the unclosed file.

There is no simple manner in which the lost data can be recovered and it is also difficult to test the disk before writing any information out to it without knowing the exact size of the required file, the simplest solution is to keep a check on

the amount of free space with a catalogue request and start a new disk when space starts to run short.

2. **Block/Memory Commands Destroys Error Information.**

Since the block and memory commands use the error channel for communicating with DOS the last information in the error channel is always destroyed. This is not a bug, any error checks to and from the disk should always be made straight after the disk command has been given. On the 4000/8000 series use DS for the error code and D\$ for the error message.

3. **Calling The Directory From Both Drives.**

If no drive is specified then the DOS returns the directory from the last drive which was accessed first and then the second directory. The drive from which the directory has come is shown next to the disk title.

4. **Load/Run.**

Drive zero initialises and the first file is read in. If the file is not a program an error message is generated. If the drive has already been used then the last program called up will be read into the machine. If you wish to use the quickload feature (Shift RUN/STOP) ensure the program to be run is the first on the directory.

5. **If the user sends more than 4 files to be copied from one drive to another, the DOS should return an error message, under some circumstances this does not occur and the error message 'Disk ID Mismatch' will be given. Do not send more than four lines to be copied at a time. For large amounts of copying use the disk utility 'copy disk files'.**

6. **Copying Files on the 8050.**

The copy command on the 8050 will sometimes fail after

eight copies when 'COPY DO TO D1' is given usually giving 'Illegal Track or Sector' or 'DISK ID Mismatch' as the error message. Copy also fails after eight copies when performing matched copying using 'COPY DO, "TEST*" TO D1,"*".

The copy bug is not present on DOS 2.1 as the two systems are different. (DOS 2.5 has to keep bringing in the BAM for the drives while DOS 2.1 keeps both BAM'S in DOS RAM.

7. **Block Allocate on the 8050 and 4040 Disk Systems.**

The Block Allocate (B-A) command is for use with the random access commands so that the disk unit knows that the block has been used and can not be allocated to programs or sequential files. On the 3040 and 4040 drive the BAM on the disk is updated only when a file is closed. The drive should not be initialised before the BAM is written back to the disk as this actually calls the old BAM from the disk. B-A is supposed to report via the error channel whether a block has been allocated.

If the block has not been used it reports OK but if it has then the message also reports the next free track and sector. On the 3040/4040 this works correctly. On the 8050 however there is a bug in DOS which can cause problems. For example if the whole of track one has been used then DOS will report 'NO BLOCK 1 21', this is obviously incorrect in that there is no sector 21, i.e., DOS reports an illegal sector and will not search other tracks. Thus if a program requires the next block it has to perform a track by track search and test for illegal sectors to find out when it has found a free block.

8. **Disk Initialisation on DOS 2.5.**

There is no problem with the 8050 disk system at power on.

The bug has the appearance of being sporadic but is very repeatable when the cause is known. For instance the disk may not initialise, programs will not be found or the directory will not be displayed. The problem is caused by having the read head above track 55 at power on. The disk system can not recognise the problem and tries to read information from the track that it is on which causes considerable confusion.

In a business system where a shift run is the method of getting the program running it is important that the head be moved to around track 39. This can be done during the system power down at the end of the previous day by performing an initialise via the error channel, the disk will then be in the correct state for use the next day. Note that the problem only occurs when the disk is powered down and then up again.

9. Disk Initialisation DOS 2.1.

DOS 2.1 has an auto-initialise routine which relies on the disk ID changing. Thus it is important for the user to ensure that ID's are different between disks for that the initialise command is given via the error channel. DOS 2.5 has a microswitch which will initialise the disk whenever it is opened and closed.

10. Block Write.

This command should not be used under any circumstances on DOS 2.1 as it clogs up the error channel, the only solution being to reset the disk by power down. Use the function U2 instead.

11. Disk Status DS\$.

The disk error string does not monitor the commands which utilise the error channel (15), the commands associated with this are all the low level block and memory commands such as Block-Read. The solution is to read the error from disk using:—

```
10 INPUT #15,A$,B$,C$,D$ :
PRINT A$,B$,C$,D$
```

12. Channel 14. There is no channel 14 in DOS 2.x, use only those between 2 - 13 for normal

random access. Secondary addresses 0 and 1 are used for LOAD and SAVE and should only be used under special circumstances.

Basic 4.0 String Bug.

The bug is a failure to detect and report an 'Out of Memory' error. This can cause corruption of string data or programs if space is running short.

The bug only occurs in BASIC 4.0 and when there are less than 768 bytes free after all variables and arrays have been assigned by a program.

An example of the bug on a 32k PET

```
10 DIM A(6330)
```

```
20 BUG$=BUG$+"W"+".".
PRINT BUG$: GOTO 20
```

The above program should concatenate a string of alternating characters 'W.W.W.W.W.'. The 'Out of Memory' termination is correct but the string is corrupted after only a few passes.

Solution.

```
1. Direct from BASIC
IF FRE(0)<768 THEN
PRINT"OUT OF MEMORY":
STOP
```

Converting Programs to run on the 4022 Printer.

There are only two secondary addresses effected by the change from 3022 to 4022.

Secondary Addresses 3

- Lines/Pages.

On the 3022 the syntax for this command would be:—

```
OPEN 3,4,3, PRINT#3,60:
CLOSE 3
```

If this command is sent to the 4022 it has no effect on the page length. On the 4022 the syntax for the command is:—

```
OPEN 3,4,3, PRINT#3,CHR$(60):
CLOSE 3
```

Unfortunately if this command is sent to a 3022 the page length will not be altered. Thus it is necessary to use something such as the following:—

```
10 OPEN 3,4,3: PT=0:INPUT
"PRINTER TYPE 3022/4022":A$
20 IF LEFT$(A$,1)="4" THEN
PT=1
1000 PRINT#3,NL: IF PT THEN
PRINT#3, CHR$(NL)
```

Secondary Address 6

- Lines/Inch.

On the 3022 the default value is 24 at power on. This gives 6 lines/inch. For closing the text up so that graphic characters join, it is necessary to send 18 to the printer.

On the 4022 this printer will default to 36 at power on giving 6 lines/inch. 8 lines/inch is achieved by sending 24.

To maintain compatability between 3022/4022 the following could be used:—

```
1000 PRINT#6, CHR$(18): IF PT
THEN PRINT#6, CHR$(24)
```

Note also that the 4022 printer will not accept a line spacing greater than 127.

Other Differences.

The 4022 printer is obviously slower than the 3022 but a time saving can be made if the fact that the matrix head actually returns to the start of the line rather than moving to the end of the line as with the 3022. This is especially useful in processes such as screen dumps.

CHR\$(141) which returns the carriage to the start of the line without giving a line feed increments the page counter on the 3022 which will obviously knock out the paging option. This has been corrected on the 4022 so than only a line feed causes the page counter to increment.

If 0 is sent to format sorting 999 a zero will be printed on the 4022 but it will be left blank on the 3022.

The 3022 does not recognise CHR\$(12) as a form feed, the 4022 will action this correctly.

The minimum page length on the 3022 is 10 lines/page. On the 4022 the minimum is 5 lines/page.

The 4022 and Commodore Software.

The 4022 printer works with the following Commodore packages when the ASCII option is selected:—

The Accountant

Paymaster

Select PET graphics option on OZZ to run it with the 4022.

As yet the 4022 does not work with Company but ISA have a 4022 and are investigating the necessary changes.

Dave Middleton
Software Technical Support

The 8026 Printer

The 8026 printer is essentially the same as the original 8026 printer which has been in use for a few months.

The printer has a new interface which makes it less intelligent than before. It is no longer possible to select a line feed by means of a dip switch inside the printer, line feed is automatic. Software should not be modified to enable underline using ASCII 15. It is necessary to use backspace and underline (ASCII 223 on the printwheel - Olympia 80110 758). This will obviously mean that most of the software now in use will not be able to run correctly on the printer.

WordPro4 runs but many of the formatting features are not available. WordPro4 has to access the printer as a CBM device. The spinwriter which is also available as an option in WordPro4 causes garbage to be printed on the 8026 indicating that the control codes are not the same.

WordCraft80 runs with the same limitations as WordPro by selecting the 3022 printer option but for some reason output will be at 1.5 line spacing.

The following are items apparently missing from the printer:—

1. Form feed - There is no way to move quickly to the top of the next form, it is necessary to keep a line count within the computer and then skip lines at page boundaries.
2. Underlining - It is not possible to underline directly, i.e., ASCII 15 on the original 8026 printer underlined. It is necessary to send backspace followed by underline.
3. Bi-directional printing - With a print speed of only 17 cps it could be thought that bi-directional printing or even logic seeking would increase throughput but when the printer is in free flight the carriage moves so fast that the extra logic required by the printer is not necessary, the overall printing time would only alter slightly.
4. Carriage return with no line feed. ASCII 141 is usually used to send this allowing overprin-

ting. This is not possible on the 8026, if overprinting is to be performed it is necessary to backspace to the correct position.

5. There is no provision for altering the number of lines/inch from software, there is a toggle switch which allows 1, 1.5 and 2 line spacing. It is possible to make the printer output at 1.5 line spacing by printing to a logical file number greater than 127, e.g., OPEN 128,4: FOR A=1 TO 10: PRINT #128, "AAA" NEXT

Conclusions

The 8026 printer offers the user a very simple ASCII printer. It scores against many other printers, including the 8027, by doubling as a standard typewriter and many of the functions on the 8027 printer are switch selectable on the 8026. The slow print speed may put some people off but the cost of the unit is going to gain many followers especially amongst people who only want to do a bit of word processing. However, as a full time word processing unit the printer would be intolerably slow unless the computer could multitask between the typist and a document spooled onto disk.

The latest version of WordPro is supposed to be able to output to a printer as well as allowing the user to continue working with the computer, i.e., spooling. Wordcraft will also need to be modified in the same manner if it is to be effective with this printer.

The 8027 Printer

The 8027 printer is essentially the same as the 8026 printer which has been in use for a few months but without the keyboard.

The printer has a new interface which makes it much more intelligent than the 8026. Like the new 8026 underline is no longer available with ASCII 15. There are 20 software control codes available which makes the printer very powerful.

WordPro4 runs but most of the formatting features are not available, i.e., underline. WordPro4 has to access the printer as a CBM device. The spinwriter which is also available as an option in WordPro4 causes garbage to be printed on the 8027 indicating that the control codes are not

the same.

WordCraft80 runs with the same limitations as WordPro4 by selecting 3022 printer option.

The 8027 has two very powerful features which are not available on many other daisy wheel printers:—

1. The pitch of the output characters is totally under the control of the user, this allows any pitch daisy wheel to be used or allows the user to select the characters/inch to bring attention to titles, e.t.c.,

Most useful is the programmable mode whereby a character is sent to the printer followed by a number which controls the inter-character spacing. Thus the user can have the true proportional spacing.

2. The user also has total control over the movement of the carriage and platen, and whilst the printing speed is only 17 cps when absolute horizontal and vertical addressing are used the head will move about very fast, possibly about 150 cps. Control over the movement is to the nearest 0.5mm and careful programming can give output very similar to that of a digital X/Y plotter, with very high accuracy. The following are items apparently missing from the printer:—

1. Form feed - The only way to move quickly to the top of the next form is to perform a calculation within the computer and send the results to the printer as a vertical tab.
2. Underlining - It is not possible to underline directly, i.e., ASCII 15 on the original 8026 printer underlined. It is necessary to send backspace followed by underline.
3. Bi-directional printing - With a print speed of only 17 cps it could be thought that bi-directional printing or even logic seeking would increase throughput but when the printer is in free flight the carriage moves so fast that the extra logic required by the printer is not necessary, the overall printing time would only alter slightly. The printer can have

the carriage moving from right to left but all this does is print the text backwards.

4. Carriage return with no line feed - ASCII 141 is usually used to send this, allowing overprinting. If the 8027 is in auto-line feed mode it is necessary to toggle this off, send ASCII 13 and then toggle auto-line feed back on again. Alternatively back space could be used.

Conclusions.

The 8027 printer offers the user a very powerful ASCII printer. The slow print speed may put some people off but the cost of the unit is going to gain many followers especially amongst people who only want to do a bit of word processing. However, as a full time word processing unit the printer would be intolerably slow unless the computer could multitask between the typist and a document spooled onto disk.

Packages which have been altered to run on the 8026 should run without much trouble on the 8027, one of the big problems will be in using the auto-line feed. On the original 8026 this was switch selectable but on the 8027 this is toggled on and off by the same control code, thus if a program is run repeatedly output will alternate from being correctly spaced to a single dark line!

The latest version of WordPro is supposed to be able to output to a printer as well as allowing the user to continue working with the computer i.e., spooling. WordCraft will also need to be modified in the same manner if it is to be effective with this printer.

As the 8027 is an intelligent peripheral it will need software to be written or modified and until this is done it will be looked upon as being very slow, standard ASCII printer, a great loss considering the features available.

Programming the 8027

Printer:-

Refer to figure 1.

Note 1:-

The current position of the head is taken as the tab address.

Note 2:-

The third byte indicates the number of lines to tab.

Hex Code	Action
08	Backspace
0d	Return
0a	Line Feed
1b 0a	Horizontal Tab Right
1b 0a	Horizontal Tab Left
1b 07	1/2 Line Feed Negative
1b 03	1/2 Line Feed Positive
1b 04	Line Feed Negative
1b 0b	Margin Release
07	Bell
1b 01	Set Horizontal Tab (Note 1)
1b 02	Clear All Tabs
1b 08	Selective Tab Clear (Note 1)
1b 09	Set Left Margin (Note 1)
1b 00	Set Right Margin (Note 1)
1b 0c nn	Define Vertical Tab (Note 2)
1b 0d nn	Pitch (Note 3)
1b 05	Move Carriage Left to Right
1b 06	Move Carriage Right to Left
1b 0e	Programmable Mode (Note 4)
1b 0f	Exit from Programmable Mode
7f	Correction Marker (?)
1b 6c	Clear Correction Marker
1b 4n nn	Vertical Tab Negative (Note 5)
1b 2n nn	Vertical Tab Positive (Note 5)
1b 40 00	Fast Platen Move Negative (00 at end?)
1b 20 00	Fast Platen Move Positive (00 at end?)
11	Switch off Auto Line Feed
1b 3n nn	Horizontal Tab to Absolute Address
1b 10	Bold on/off
1b 11	Double Printing on/off
1b 50	Auto Line Feed on/off
1b 12	95 Print Wheel
1b 13	96 Print Wheel

Figure 1.

Note 3:-

This is the inter-character spacing, it would seem that 6 gives 10cpi and 12cpi.

Note 4:-

This allows proportional spacing to be used. When switched on the ASCII character is sent followed by the required spacing.

Note 5:-

This allows a range of 0 to 4096 increments of approximately 0.5mm. If a vertical tab has been set and 0 is sent the platen moves to the required location.

Relative Records Bug

There is a serious bug in the relative record system on both DOS 2.1 and 2.5. The bug only occurs

when two files have been opened for reading and writing. The bug only shows at certain length records and at set distances through the file. The following example demonstrates the bug, (figure 2).

The program sets up 2 files (30 - 250) with unique records. The first 34 records are read from 'Keytest' then a record is read from 'Filetest'. Now records on 'Keytest' are updated. Both files are then closed (280 - 450). When 'Keytest' is read again some of the updated records are unchanged, in this example records 34 - 40 are the same as they were originally.

Thus it is not possible to have two relative files open for reading/writing at the same time with any degree of certainty that records will be updated correctly.

1. Open and close each file before accessing another.
2. Thoroughly test the record length chosen to see that it does not cause the bug.
3. This solution has no reason for working but it cured the bug in the example program so try it at your own risk: When the files are opened in lines 280 and 290, position the record pointer at record number 1, read it into the PET, reset the record pointer to 1 and then write it out again. The file then reads and updates correctly. Do this for both files.

There are potential software problems arising from the recent introduction of the 12" screen 4032 and the impending introduction of the 12" screen 4016.

Both the 9" and the 12" screen machines use £97 (decimal 151) to hold a value related to the key being depressed. In the 9" 4016/32 this value relates to the keyboard matrix: in the 12" 4016/32 it is pseudo-ASCII (i.e. the 9" is like the 3032, while the 12" is like the 8032), see figure 3.

To check whether a program uses PEEK(151) use the FIND command in the BASIC Programmers Toolkit or Basic Aid.

40-column machines use 224-248 (£E0-F8) for a table of screen wrap-round flag. Because such a table became unnecessary in 80-column machines, these addresses were used for other purposes (e.g. bell, repeat, keyboard buffer length). The 12" 4016/32 has these 8032 features, but needs the table of wrap-round flags. Therefore the 12" 4016/32 uses even more of the top of the second cassette buffer than does the 8032. No programs should now use addresses between 1001 and 1024 (£03E9-0400). (See figure 4).

```

30 DOPEN#1, "KEYTEST", L13, DO
40 FOR J=11 TO 50 - 100
50 A$=STR$(J)+"++++++":A$---MID$(A$,2,13)
70 RECORD#1,(J)
80 PRINT#1,A$
90 NEXT
100 DCLOSE#1
110 DOPEN#2, "FILETEST", L254, DO
120 FOR J=1 TO 50
130 B$=STR$(J)+"++++++"
140 RECORD#2,(J):PRINT#2,B$:NEXT
150 DCLOSE#2
190 DOPEN#1, "KEYTEST", L13, DO
200 FOR J=1 TO 50:INPUT#1,A$:PRINTA$:NEXT
210 DCLOSE#1
220 DOPEN#2, "FILETEST", L254, DO
230 FOR J=1 TO 50: INPUT#2,A$:PRINTA$:NEXT
240 DCLOSE#2
250 PRINT"PRESS A KEY
260 GETZ$:IFZ$=""GOTO260
280 DOPEN#1, "KEYTEST", L13, DO
290 DOPEN#2, "FILETEST", L254, DO
300 X=34
310 FOR J=1 TO X: INPUT#1,A$
320 PRINTA$: NEXT
330 RECORD#2,25
340 INPUT#2,B$
350 PRINTB$
370 FOR J=X TO 50
380 A$=LEFT$(A$,9)+"TEST"
390 RECORD#1,(J)
400 PRINT#1,A$
410 PRINTA$
420 INPUT#1,A$
430 NEXT
440 DCLOSE#1
450 DCLOSE#2
510 DOPEN#1, "KEYTEST", L13, DO
520 FOR J=1 TO 50: INPUT#1,A$:PRINTA$: NEXT
530 DCLOSE#1

```

Figure 2

4016/32 Keyboard layout														<i>Figure</i>				
15	80	72	79	71	78	70	77	69	76	68	75	7	14		74	66	73	65
Q	'	"	#	£	%	,	&		()								
64	33	34	35	36	37	39	38	92	40	41	95	91	93		19	17	29	20
8	64	56	63	55	62	54	61	53	60	52	59	5	12		58	50	57	49
	Q	W	E	R	T	Y	U	I	O	P		<	>		7	8	9	/
18	81	87	69	82	84	89	85	73	79	80	94	60	62		55	56	57	47
48	40	47	39	46	38	45	37	44	36			27			42	34	41	33
	A	S	D	F	G	H	J	K	L	:					4	5	6	*
65	83	68	70	71	72	74	75	76	58			13			52	53	54	42
32	24	31	23	30	22	29	21	28	20						26	18	25	17
	Z	X	C	V	B	N	M	,	;'	?					1	2	3	+
90	88	67	86	66	78	77	44	59	63						49	50	51	43
					6										10	2	9	1
															0	-	=	
					32										48	46	45	60

Figure 3

```

12" 4016/32      8032
03E9 1001      00E6 230      Repeat Key Delay Counter
03EA 1002      00E5 229      Delay between Repeat Counts
03EB 1003      00E3 227      Maximum Keyboard Buffer Size
03EC 1004      00E7 231      Bell Enable and Delay Count
03ED 1005      50      50 Herz Jiffy 5 Counter
03EE 1006      00E4 228      Repeat Key Flag
03EF 1007      Mask for Tabs
03F0 1008      )
03F1 1009      )
03F2 1010      )
03F3 1011      )
03F4 1012      )      10 bytes for Tabs
03F5 1013      )
03F6 1014      )
03F7 1015      )
03F8 1016      )
03F9 1017      )
03FA 1018      User command for Monitor
03FB 1019
03FC 1020      Timeout on IEEE
03FD 1021
03FE 1022
03FF 1023
0400 1024      0400 1024      Start of Basic

```

Figure 4.
Memory Map
for top of se-
cond cassette
buffer in 12"
4016/4032



There are (at time of writing) three general styles of PET/CBM software ROMs. ROM stands for Read Only Memory - their programs are set at the factory and cannot be changed. That's OK - few of us have the talent and/or ambition to change the internal workings of our PET/CBM system, and even if we did we'd lose one of the great advantages of the home computer: the ability to exchange programs with others. The ROM programs, being pre-written and burned in, are there the moment we turn the power on.

I call the three generations of Basic: Original, Upgrade, and 4.0. The first two generations had confusing number system: some Commodore divisions called Original ROM sets Basic 1.0; others called the same thing Level 2 Basic. When the Upgrade system arrived, the numbers changed to Basic 2.0 and Level 3 so as to make the confusion one hundred per cent. By the time 4.0 Basic came along everybody synchronized, and the machine prints BASIC 4.0 to end

the problem once and for all.

Within each Basic version, there are small differences to accommodate variations in the hardware. The Business or ASCII keyboard - the one with numbers across the top row - needs to be scanned in a different way than the graphics keyboard; and 80 column screens must be worked in a style that differs from the 40 column display. These differences are reflected by changing one ROM out of the set to allow for the configuration desired. The other ROMs in the set are the same regardless of hardware.

The first Basic came as part of the Original ROM set. It had a lot of limitations. You couldn't put more than 256 items into an array; you couldn't do a successful IEEE-488 input; you had no machine language monitor; tape data files had potential problems. Most users breathed a sigh of relief when the newer Upgrade system became available.

If you have Original ROM Basic, your PET will power up with the

message: *** COMMODORE BASIC ***. Note the asterisks: they are the signal that you have the Original system.

If you have this early Basic, it's worth while thinking about moving up by obtaining a replacement ROM set. You'll get technical benefits. More important, you'll be joining the mainstream of PET/CBM users and be better equipped to exchange ideas and programs.

Upgrade ROM solves the above limitations. Users with Upgrade ROM will see ### COMMODORE BASIC ## when they turn the power on. Whether you call those symbols pound signs, number symbols or hash marks they clearly flag Upgrade ROM.

Users with original hardware can refit their machines to Upgrade ROM, but they will have trouble in taking the next step to 4.0. It's not just that they are missing the ROM socketing to plug everything in: they would also find that screen "noise" would start to give trouble. Basic 4.0 doesn't politely wait for the screen to be ready before delivering new information ... characters are slapped in at full speed. Newer machines won't see any problem, but the original boards may end up with a screen that looks like a snowstorm.

The newest Basic so far is 4.0 and it's easy to spot: the screen announces the number. The changes here are useful, but not essential. You get new commands for disk: things like DLOAD, CATALOG and SCRATCH. You get somewhat better file handling; and that great time-waster, garbage collection, has been speeded up so that it is no longer annoying. In many respects, the 4.0 improvements are largely cosmetic; they support ease of use rather than eliminating fundamental road blocks.

There's great compatibility between the various versions of Basic, especially between Upgrade and 4.0 ROMs. Each user tends to exploit the features he is given, however, so that programs on a more recent model may not be able to time-travel back to earlier versions.

Still, they are all PETs. They all have that style ...

When the 'phone rings in my office it usually means that somebody is having problems with programming a PET, or the machine is not giving the desired response. To be asked "Have you a current passport" is slightly disconcerting. However, since 'passport' and 'travel' are the same thing I spend the next few minutes saying "Yes" quite a lot....on friday I was going to France!

Martin Maynard, who runs Audiogenic (see the back cover of any club news) is trying to get his private pilot's license, so what better way to learn than to take a small plane over to France to conduct some business with Procep, Commodore's French distributor.

Having never flown in such a small plane I was quite apprehensive as to how comfortable the flight was going to be (brown paper bags and green faces come to mind), so sitting in the 'plane at Denham airfield waiting to take off was probably the worst moment of the flight. In the event however the takeoff was smoother than in many commercial jets, quite a surprise! We were quite lucky to have good weather - a small 'plane is affected quite badly when flying through cloud.

The first landing was at Southend to refuel and pick up the duty free, which incidentally is a lot cheaper than normal airport shops - £3.50 for a bottle of Southern Comfort. (Ed. note - I knew there was some reason for him going there!).

The next stop was Le Bourger. The airport is right next to the Charles De Gaulle 'port, and we followed the main flight path in over the airport, turned left at the end of the runway, and landed a couple of miles further on at Le Bourger.

The taxi drive into Paris was more hair raising than the flight! French drivers do not seem to mind which side of the road they're on. Later on when we went out for lunch I found out that pedestrians lead even more dangerous lives. Pedestrian crossings are fairly common, but motorists take almost no notice of them at all. Step onto a crossing and there is a good chance that the gutter is where you will wake up. The only way to cross

the road is by teamwork. If we all rushed out together they usually stopped. They seem to draw the line at driving on the pavement!

We were on a fairly tight schedule and could only spend a couple of hours at Procep, because we had to be back at Enfield before it was dark ; there being no landing lights.

In the event the schedule was a bit out and it was twilight before we landed at Stanstead for passport checks. Many a World War II fighter pilot would have been envious of the speed with which we landed and took off again. Seven minutes in all, including a delay in waking the customs official!

Landing at Enfield was fairly exciting because it was now completely dark. Navigating by roadsigns must be one of the more difficult ways of travelling. The landing was, however, probably the best one on the whole trip - the pilot's response was "When you have to get it right....you get it right".

And what, in the end, did we see ?

EDEX. This is a Toolkit-like chip with the now normal auto line number, renumber, delete, find, variable dump, trace and error display. At this point similarity ends. Edex has two powerful commands which can change the way programs are written. These are :- IF...THEN...ELSE..., and PRINT USING

Most programmers will have cursed the lack of an 'ELSE' facility in Microsoft Basic because it can simplify program generation and also make code more readable. The syntax for 'ELSE' is :-
IF EXPRESSION THEN
TRUE EXPRESSION ELSE
FALSE EXPRESSION
E.G.

A=5:IF A=5 THEN PRINT "A IS FIVE":ELSE PRINT "A IS NOT FIVE"

Another good feature is that the 'ELSE' does not have to be on the same line as the IF...THEN. E.G.

5 A=10 : IF A=5 THEN PRINT "A=5"

10 ELSE PRINT "A IS NOT 5"

'ELSE' will not alter the program flow either. E.G.

5 A=3 : B=7

10 IF B=5 THEN PRINT "B=5"

20 IF A=3 THEN PRINT "A=3"

30 ELSE PRINT "A IS NOT 3"

40 ELSE PRINT "B IS NOT FIVE"

When this is run the following will be printed :-

A=3

B IS NOT FIVE

If this was being treated correctly, lines 20 and 30 would be skipped as they form part of the 'TRUE' statement. Thus the program should only print :-

B IS NOT FIVE

PRINT USING

This allows the user to format data in the same manner as with format controls on the 3022/4022/8024 printers. There are a few differences, so a format string can not be used as a format string for the printer (but see later), but what's the point anyway ? Data can be sent to other printers as easily as it can to the screen, e.g. :-

10 DN=3 : OPEN1,DN

20 PRINT USING #1, A\$,D,C,D

30 CLOSE1

Change DN to 4 and output goes to the printer!

Back to format strings. This only applies when using secondary address 2. A typical format string looks like :-

A\$="/ /AAAAAAA,/ /9999.99/"

PRINT USING A\$,B\$,C

BONJOUR 5201.53

The slash is used to indicate the start and finish of character strings.

The last command which is useful in many situations is quarter square plotting, giving 79 by 49 on a 40 column and 159 by 49 on an 80 column machine. Points are set using PLOT X,Y and RESET X,Y to clear a point.

MULTEX. This is a chip which replaces the current \$F000 - \$FFF ROM. The IEEE routines which reside in this ROM have been modified, allowing a number of PETs to talk with the same peripheral at the same time.

Multex allows two PETs to have two sequential files open at the same time. This is not possible with MUPET. Multex uses a random delay before accessing the IEEE bus

continued on page 19.



VIC holding its own!

Commodore Splits the Proton

Well, not quite, but we did help the B.B.C. out with a PET a few weeks ago. The story started in a deceptively simple way.

Called into a meeting with Applications Manager Clive Booth and Marketing Manager Keith Hall, we thought we were going to discuss some future newsletter projects. In his inimitable little way, Keith said "Just before we start, an interesting project for you Peter. How would you like to spend the day with the B.B.C., helping out in a science fiction serial they're doing?" Fine, I thought - day out of the office, interesting day's filming, never having seen the B.B.C. in action before.

"Just a few snags though" said Keith. "Shooting starts on Friday" (this was on Tuesday), "they need an 8032 with Greek Multifont, and they want to start at 7.30 in the morning. Can I leave you to sort it out? Thanks."

And so into the meeting, with me thinking "How do we editors get involved in such things?,"

In the end I contacted fellow drinking companion Chris Palmer, of Adda

Computers Ltd., who happen to have a store in Ealing where the B.B.C. were going to be doing the day's filming, and who also happen to be a distributor for Multifont. After promising several pints of Fullers a deal was agreed upon, and we decided to meet up at Adda at seven thirty on Friday morning.

Then came the next problem, namely getting to Adda at seven thirty in the morning. I live in Taplow, a village not renowned for its frequent train service (stage coaches have been rumoured to have been seen), and for me to get the appropriate train into Ealing meant getting up at six o'clock in the morning. The only time I ever see six o'clock is during an all night party, and the type of party I go to usually means that I don't see it very clearly even then. So, my trusty alarm clock came into action (which was more than I managed to do), and even survived being hurled against the wall when it went off on the great day.

And so to Adda, Chris Palmer, looking if possible even more bleary eyed than me, was waiting with an 8032, a cassette deck and a Greek Multifont. I stood back in amaze-

ment, awe even, at this great preparation. A fag and a cup of coffee later and we set off with a taxi driver who managed to do a very good impersonation of Stirling Moss in his heyday. Still, we arrived intact and found ourselves being shown into a room without a power socket. After we'd pointed out one disadvantage of modern computers, i.e. they don't work very well without power, we were shown into another room and allowed to get on with it.

Did the B.B.C. know what they were letting themselves in for? The basic story, or at least the bit of it we were involved with, revolved around a school and one pupil in particular, who in conversing (via the keyboard) with the PET discovered himself to be a mathematical genius and could produce great complex formulae with no apparent effort. The computer duly responds with further problems for him, and he succeeds in solving them all. Hence the need for Greek Multifont, which enabled us to display the required goodies on the screen.

After working for an impressive ten minutes (we were impressed anyway) we were told it was breakfast time,

Educational Supplement

Welcome to this, the third edition of our special educational pull-outs. I hope you've enjoyed the two so far, and will continue to do so as time goes by. One thing I do need though, is more contribution from you, the educational user. I'm sure a lot of you are doing all kinds of interesting things with PETs, and I'm equally sure that there are plenty of other people who would be interested in reading about it. There's no limit to what I put into this section - we can take over the whole magazine if you want! - so let the world see what you're doing.

Every week people in schools are encountering PETs for the first time, and it is of obvious value to them if they can have as much information as possible on how others have got on with PETs. Let's use the medium of this educational pull-out to pass on past experiences to new (and not so new!) users, and help them out.

And it's to your benefit as well, not just theirs. The more schools we have with PETs, the bigger base of software that we get, and so the number of good programs grows and grows as we establish Commodore as the company to go with. There will always be a need for more, good, educational software, and this is one way of, if nothing else, getting more programs coming in. So get going, and send those articles in. If nothing else, it's free advertising for you!

Educational Workshops

No amendments to the list of educational workshops this time (but one or two new ones), which means one of two things. Either we got it completely right last time (which I can't believe!) or you're not paying attention. Let us know if we have made any mistakes, and I'll publish corrections next time around. You'll notice if you look at the lists in issues 4 and 6 of this current volume, that some areas have loads of workshops, whilst many have next to none. If you're in such an area, and would like to help out other schools, colleges etc. that are around about (and receive an awful lot of software and regular mail-outs free of charge) contact Jean Frost at the new address in the magazine editorial for an application form.

Workshops are vital - they're the real human link for schools thinking about entering the PET field, who need a place to go to for help and advice. And as I mentioned above, the benefits are not only to them. So far

we've distributed over 300 programs free of charge to the workshops - a vital contribution to the educational arena, where costs are always at a premium and anything is welcome. In total (at the time of going to press, as they say) we've mailed out over 90,000 programs for free. Can't be bad.

VIC Programs

I've included in this section this time a selection of VIC programs. Whilst realising that not many of you will actually have VICs as yet, one of the great strengths of any micro is the software that supports it, and by publishing programs now we'll be helping you to get started as soon as you do begin getting VICs.

What I've featured in particular is a whole collection of stuff from Paul Higginbottom, well known PET genius. Paul was given a VIC to explore the inner workings of it, and inside we give his findings. Lots of fascinating information that you won't find elsewhere, and that will certainly save you an awful lot of time when you start your own delvings into the VIC world.

I'll endeavour to publish as much of this kind of material as possible as more of it becomes available, and keep you informed of VIC developments generally as well. As the newsletter regularly contains sections devoted to BASIC and machine code programs for the PET, and specials on programming a particular peripheral (disk drives in the last two issues, cassette decks in this one), so the educational section will contain similar features on VIC. We'll make this the definitive place to go for VIC information of all kinds.

One person who will in the future be regularly contributing to our VIC section will be Malcolm North of Commodore, who has recently rejoined us after a spell in the outside world

with one of our dealers. He has come back especially to be working on the VIC, and will be keeping us informed of all new developments as and when they happen. Sample programs will be a feature of Malcolm's articles, as we try and keep you as up to date as possible on VIC news.

Educational games

I've hinted briefly elsewhere in this newsletter at the importance that games have in the educational world. Many people, rather unfairly I feel, decry games on computers, and indeed after issue 4 I received a very stern letter from somebody who shall remain nameless about the complete and utter waste of two pages of the newsletter! Well, everyone's entitled to their views of course, but if games can overcome a child's fear of the micro, then they should not be written off.

Many of the better educational programs, e.g. Russell Wills series of programs that are distributed by Audiogenic (0753 595269), or the excellent programs available from Supersoft (01-861 1166), cleverly combine learning with game playing, so that quite often the child doesn't even realise he's learning whilst enjoying himself in front of the friendly micro. So don't write off the games, as the article inside goes into in further detail.

There is a large number of educational programs on the market now. Going back to the workshops for a moment they have around 300 programs each now, which they were given free of charge by Commodore under the public domain software scheme, and many of the workshops have an awful lot more software of their own. It is a ground base of software like this that should ensure the PET stays up at the top in education for a long time to come - no other micro has so much software going for it, and it would take a very long time for some new machine to even begin to catch up with it.

Not so with the VIC however. Despite VIC Computing's somewhat extravagant claim that the VIC will run all software that is available, this is not quite the case. However, with

not too much modification (screen format generally, as most of the educational programs around are written totally in BASIC, and VIC operates a BASIC very close to the heart of BASIC 2) all of the existing software will quite happily be transportable. So even before the VIC arrives in vast quantities it will have the necessary software ready and waiting.

And then of course there's the new software being developed especially for the VIC itself. Programs similar to the enormously popular Strathclyde Basic are coming along, and will no doubt sell in vast quantities when they appear. The future's looking good.

Staying with VICs, there have been complaints about it only having 22 columns across the screen. Well, having seen the VIC arcade games, I have no fears. Using the definable character function it really does seem that the VIC has high resolution graphics built in. And in normal mode the fact that a screen display can be seen by the whole classroom without lots of eye straining is a big plus.

What the papers say

The new feature in the other section of the newsletter, namely the "What the papers say" article, should be of interest to you as well, as it will be covering all kinds of PET mentions in the press, whatever the area of application. Find out what the rest of the world is doing with Commodore hardware - you'll be as amazed as I was.

Educational Editorial

Anyway, read on and see what's in store this month. Lots of meaty articles interlaced with the usual mixture of goodies on programming hints and tips and so on. See you again soon!



COMMODORE INTERNATIONAL DONATES SIX PET 2001 R COMPUTERS TO UNITED NATIONS INTERNATIONAL SCHOOL

Commodore International Limited, a leading manufacturer of computer systems and consumer-related semiconductor products, has donated six of its CBM/PET Computer Systems to the United Nations International School.

At a ceremony held at the School in New York City, Mr. Irving Gould, Chairman of the Board of Commodore, presented the systems to Mrs. Murray Fuhrman, Special Representative of the Secretary-General for the United Nations International School, Mr. Robert Belle-Isle, Director of the school, and Dr.

Thomas Szell, head of the science department.

In accepting the CBM/PET Computers, Mrs. Fuhrman said, "We are very appreciative for Commodore's donation. The Commodore PET is the school's strong preference because of its outstanding graphics with full mixed text, its speed, compactness of design, and particularly because of its wide acceptability and utilization by educational institutions worldwide. These Commodore Computers are a most important and welcome addition to the United Nations International School."

Mr. Gould commented, "Commodore International is honoured to contribute to the U.N. School's educational curriculum. This school represents an important effort to break through national and cultural barriers in education. As a multinational computer company and a pioneer in the development of classroom computer systems, Commodore believes it is important to support the United Nations School."

Commodore International has more than 110,000 computers in use around the world, and is a dominant manufacturer of small computers used in the classroom, in business, science and industry.



Mrs. Murray Fuhrman, Special Representative to the Secretary General, accepted one of several CBM/PET computer systems donated to the United Nations International School by Commodore International. Commodore was represented by Irving Gould, chairman of the board (left), with Mr. Robert Belle-Isle, dean of the school, and Dr. Thomas Szell, science department chairman, looking on.

(The school had specifically requested Commodore computer systems for use in the classroom, on the basis of the amount of educational software available, the IEEE interface which permits instrumentation/lab work, graphics and other features. Commodore will be providing an introductory computer seminar to the teachers when the school resumes full time in the fall.)

Pet in Education Conference

Another education conference ? At the end of November the first conference devoted to the educational uses of the PET is being held in Kensington.

Whilst recognising the need for program portability and 'the ecumenical movement' it remains true that computing ultimately depends on the machine you have and its hardware. At the session on educational hardware there will be discussions and demonstrations of PET Net, a scheme being developed by Commodore to allow interchange of programs by telephone and a central computer, and of Prestel which can now be linked directly to a PET. The hardware used is also of major importance for using the micro to control experiments in the practical laboratory.

A session on how to write CAL programs also looks valuable for PET users. As well as talks on graphics routines and machine code uses there

will be demonstrations of the frame handling programs used for author-systems. Other sessions will consider the use of PETs for schools administration, in special education and on the increasingly complex policies for microcomputers.

Our readership survey last January showed that half of all the machines in educational use are PETs, perhaps this conference will at last show the real penetration of micros in education.

The Conference is about using Commodore microcomputers in education. About frame-handling, laboratory experiments and graphics. About developing the school curriculum, the role of education authorities and about special education. Even about school's administration. Computing depends ultimately on the machine you have and its hardware. There will be discussions and demonstrations of PetNet and of Prestel.

Pet in Education Conference

We have just obtained details of some of the people who will be speaking at these conferences, and the names alone should be a good reason for going. People like Borge Christensen, David Burghes of Exeter University, Trevor Lusty (past contributor to the educational supplements, now at Starchley Upper School), Bob Lewis, of the Chelsea Science Project, Nick Hampshire, Danny Doyle, David Parkinson, of the Modern Tutorial Project, plus many more. Quite a line up!

Provisional programme.

- Friday 27: 'Using CAL programs'
Public domain software. Standards for software. COMAL.
'Educational hardware' PetNet.
Prestel and videotex.
Laboratory experiments. Trade seminars.
- Saturday 28: 'Management and administration'
Software for administration in schools. Teaching teachers.
Policies for microcomputers'
National policy. Role of LEAs.
Curriculum development Conference dinner.
- Sunday 29: 'Writing CAL programs'
Frame handling. Graphics.
Machine code. Beyond PILOT.
'Special education' Subject groups.

FREE OFFER !!

We are delighted to announce another FREE offer to educational users. One of the most popular Commodore machines in the educational field must be the 8K model, which really pioneered the use of the microcomputers in schools and colleges.

We have in stock at the moment a large number of manuals for this particular PET, which are surplus to our requirements. By filling in the form below and posting it to the address given, we'll send you a copy of the manual free of charge.

"Please send me one copy of the User Manual for the 8K PET."

Name
Establishment
Address
.....
.....
.....

Send the coupon to :—

Margaret Gulliford
Commodore Business Machines
675 Ajax Avenue
Trading Estate
Slough
Berkshire

Registration.

Applications are invited for registration and for titles of papers (15 min) or demonstrations. Early application is advised as numbers are limited to 180. Residential fee is £60, non-residential £45.

Organiser:

Dr. Christopher Smith
Department of Physiology
Queen Elizabeth College
Campden Hill Road. W8 7AH
937 5411 x 429
11 Ospringe Road
NW5 2JD 485 8632

Games in Education

A number of people claim that games have no role to play in the world of computing, and that computers should be reserved for more serious things like accounting and payroll. This is not a view that I personally adhere to, for I think that games have a good role to play in introducing people to the micro, and in schools, particularly primary schools where even the simplest game is of delight to the youngsters, that role becomes vital.

The production of educational software comes from many sources. Via the workshops is one useful medium, and most of that software comes from Canada or the States. We do not claim that that software is the best in the world, rather its aim is to give ideas and for you, the user in the real U.K. schools environment, to alter and amend the programs as you see fit. Given that those programs are all public domain, they can be copied and distributed free of charge. We would similarly expect that if you alter those programs they too would go into the public domain and be re-distributed via the workshop scheme.

Thus by giving out and receiving back programs, everybody can be seen to be benefiting from an increased software base, and as I've said before the strength of any micro is its software. So the flow continues and more programs become available.

Possibly the major source of educational software is from the schools and colleges themselves, where many teachers contribute programs they've developed themselves during, or, more usually, outside of school hours. These programs usually end up being of a high quality, since they have to go through a very exhaustive testing procedure in the school itself, and thus would get refined over the months of usage until they're honed to a fine point. Occasionally you'll get programs developed by the kids themselves, but these would tend mainly to be small programs produced during a classroom period. Undoubtedly however one or two "gems" would emerge and be used as tutorial programs.

Further Source

A further source of programs would be from commercial establishments such as Commodore, Audiogenic, and so on. These are again mainly programs written by school teachers, and tried and tested in schools and

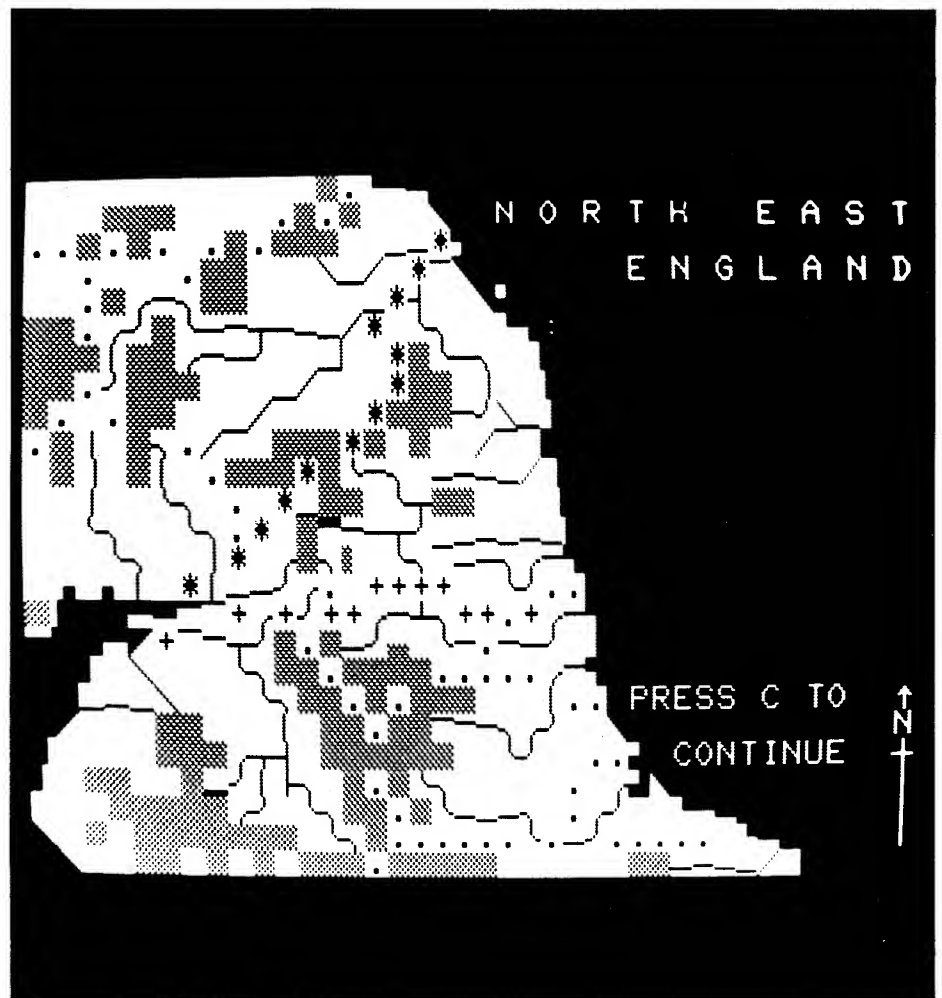
colleges etc. before being passed on to the various distributors. Whilst not decrying these programs in any way, they are obviously of limited use as they stand, having been developed at school for that school's use, and thus another school would probably have to adapt them before using them in their own environment. That is not to say they are without their uses - like workshop software they can often supply ideas for further software development.

Many programs, from whatever source, use game playing as part of their teaching process. Most, if not all, pupils seem to benefit from this - they become more willing to get to grips with the PET than if they were

using a straightforward question and answer kind of program. Obviously you'll get a far better response from someone if they're interested in what they're doing, rather than if they're just doing it because it's another part of the school curriculum.

It will also help in getting people acquainted with micros in general. They are obviously going to be of increasing importance in the future, and it's good to make sure people get the right impression of them now, so as to be prepared for their venture into the "outside world" as it were. Everybody will be using them, so it's nice to ensure that the kids who'll be using them know what they're all about.

By using games playing programs, and encouraging kids to write programs themselves (and possibly the best way to start is via games they can write and play themselves), you increase this awareness of the micro at the right age.



This piece of text is a dump of everything I have found out about the VIC-20. The model I have been working with is a US standard VIC-20 with a 3K RAM pack giving 8K total RAM 6.5K usable. I have found that this is plenty of RAM to be starting with.

What is a VIC ?

It is a low cost computer with:-
TV output to produce a 22 Column X
23 Row Screen with border
Keyboard including 8 function keys
Colour Graphics
3 voice sound output with volume
control and white noise generator
CBM BASIC interpreter version 3.0
CBM KERNAL operating system
Intelligent screen editor
High resolution multicolour graphics
RS-232
Serial Bus
Parallel User Port
Joy Stick and light pen control port
Expansion:-

Expansion Bus for:-
ROM pack boot capability
add on memory up to 32K
RAM
IEEE interface card for
peripherals
Utility cards
Etc. etc.

VIC vs PET

Commodore has done it once again moved everything around in memory that is. But hopefully this will be the last time because this is a KERNAL machine (more about the KERNAL later). The usual basic pointers, start of basic, end of text / start of variables etc. are 3 bytes higher (40-41 are now 43-44). BASIC uses zero page before \$90 and the operating system (the KERNAL) uses \$90 on. This means that BASIC is not required by a machine code program, then \$00 to \$8F can be used safely (hooray!). The keyboard buffer, and file tables etc. have all been moved up 9 bytes because the keyboard buffer is now 89 characters instead of 80. Thus the keyboard is at 631 instead of 623. The tape buffer (there's only one cassette unit allowed now) starts at 828 (\$033C). The space which used to be

the first cassette buffer is now taken up by KERNAL vectors RS-232 working storage, new flags and pointers associated with colour etc. See KERNAL listing, overleaf. The BASIC pointer I have found and the usual useful operating system variables can be seen below, the full list of operating system locations (the KERNAL), can be obtained from the KERNAL listing, figure 1 overleaf.

The KERNAL

In all Commodore computers there has been a jump table at the farthest end of ROM. The function of each entry has stayed the same (believe it or not) throughout. The KERNAL is simply an extension of this idea whereby there is a massive jump table which theoretically provide all necessary input/output routines for any device. This includes:- RS-232, TAPE, IEEE devices, SCREEN EDITOR, KEYBOARD. A lot of the jump entries in the table are indirect through page 2 of memory. Thus if a special device is being used, a module could be loaded that redirected the KERNAL vectors to specialised routines to drive this device. Hey presto! all other programs using KERNAL I/O can now drive this special device as well. The routines that are vectored are shown in figure 2 overleaf.

COLOUR on the VIC

The VIC still has all the usual video RAM which has the screen codes of each square on the screen, this starts at 7680 (\$1E00) rather than 32768 (\$8000) in the PET. However, there is also a colour nibble table which stores the colour of each character on the screen. This means that the text parser, when a line is entered or executed, doesn't have to worry about

colour at all. It's The VIC chip (thats the 6560 for NTSC US standard TV, or 6562 PAL.UK standard chip inside) uses this table when generating colour output. Each colour nibble occupies a whole byte - the top 4 bits (bits 4-7) are wasted, the first 3 bits (bits 0-2) define the colour (therefore up to 8 different colours) and bit 3 is for selection of 'multi-colour' or 'high resolution graphics' modes (these terms will be explained later). This table is addressed by the 6502 (i.e the programmer) at 38400 (\$9600) but moves to 37888 (\$9400) if the VIC has more than 8K of RAM - or more specifically if there is RAM changes from 7680 (\$1E00) to 4096 (\$1000). Right then, that's the character foreground colour table, the background and bordercolour register is next. This is at 36879 (\$900F -known as CRF which stands for chip register \$F). The first 3 bits (bits 0-2) control the border colour - thus there are 8 possible border colours. The last 4 bits (bits 4-7) control the background colours. The 4th bit (bit 3) controls 'invert mode' which if set, makes all the foreground colours become the background, and vice-versa for each square.

A facility to change the foreground colour of text that is printed or typed, is incorporated in the screen editor by pressing the CTRL (control) key and a number key from 1 to 8. These produce control characters when inside quotes or in insert mode i.e the insert key has been pressed once or more times. Thus to print "VIC" in yellow (lets suppose we have just turned on the VIC, therefore the screen is white, the output colour is blue at present, and the border is cyan), we must enter:-

```
?“ CTRL 8 VIC CTRL  
7 ” RETURN
```

Control 8 sets the output to yellow, and control 7 sets it back to blue.

The VIC responds with:-

```
?“ CTRL 8 VIC CTRL  
7 “ RETURN (in blue)  
VIC (in yellow)  
READY. (in blue)
```

Figure 1

0-2 USR FUNCTION JUMP
 43-44 POINTER TO START OF BASIC (40-41)
 45-46 POINTER TO END/OF PROGRAM/START OF
 VARIABLES (42-43)
 47-48 POINTER TO END OF VARIABLES/START OF AR-
 RAYS (44-45)
 49-50 POINTER TO END OF ARRAYS (46-47)
 51-52 POINTER TO START OF ACTIVE STRING SPACE
 (COMING DOWN) (48-49)
 53-54 POINTER TO TOP OF ACTIVE STRINGS (50-51)
 55-56 POINTER TO END OF MEMORY (52-53)
 160-162 JIFFY CLOCK (141-143)
 197 MATRIX COORDINATE OF KEY DOWN (151)
 199 REVERSE MODE FLAG 0-OFF, 18-ON
 203 MATRIX COORDINATE OF LAST KEY DOWN (166)
 204 CURSOR ENABLED FLAG 0-ON, 1-OFF (167)
 205 DELAY BEFORE CURSOR BLINKS (168)
 206 CHARACTER UNDER CURSOR
 207 CHARACTER BLINK FLAG (169)
 211 POSITION OF CURSOR ON CURRENT TEXT LINE
 (198)
 212 QUOTE MODE FLAG 0-OFF, 1-ON
 215 CONTAINS THE ASCII VALUE OF THE LAST
 KEYPRESS
 216 NUMBER OF INSERTS OUTSTANDING (220)
 220 NUMBER OF KEYPRESS IN KEYBOARD BUFR
 (158)
 243-244 POINTER TO START OF LINE CURSOR IS ON
 (196-197)
 245 FLAG FOR SHIFT/CONTROL KEYS LAST USED
 (155)
 = 255 FOR NONE
 = 159 FOR SHIFT
 = 224 FOR COMMODORE LOGO
 631-640 KEYBOARD BUFFER
 651 DELAY BEFORE REPEAT OCCURS
 652 DELAY BETWEEN REPEATS

Figure 2

\$0314	\$EABF	CINV	N/A	IRQ interrupt entry
\$0316	\$FED2	CBINV	N/A	BRK interrupt entry
\$0318	\$FEAD	NMINV	N/A	NMI interrupt entry
\$031A	\$F40A	IOPEN	\$FFC0	Open logical file
\$031C	\$F34A	ICLOSE	\$FFC3	Close logical file
\$031E	\$F2C7	ICKIN	\$FFC6	Set input device
\$0320	\$F309	ICKOUT	\$FFC9	Set output device
\$0322	\$F3F3	ICLRCH	\$FFCC	Reset default I/O
\$0324	\$F20E	IBASIN	\$FFCF	Input from device
\$0326	\$F27A	IBSOUT	\$FFD2	Output to device
\$0328	\$F770	ISTOP	\$FFE1	Test stop key
\$032A	\$F1F5	IGETIN	\$FFE4	Get from keyboard
\$032C	\$F3EF	ICLALL	\$FFE7	Close all files
\$032E	\$FED2	USRCMD	N/A	BASIC USR command vector??
\$0330	\$F549	ILOAD	\$FFD5	Load from device
\$0332	\$F685	ISAVE	\$FFD8	Save to device

SCREEN EDITOR

Each text line can be up to 88-characters (4 screen). As the last character on a screen line is typed, a line will open up below if the text line does not already occupy 4 screen lines. To see this, type the following:- (fig 1a)

Now move the cursor onto the first T in TEST and type STUPID TEST. As the I in STUPID is typed, line 20 moves down. (fig 1b)

Cursor up, cursor down, cursor left, cursor right, insert, delete, and space keys all repeat if held down. Location 650 controls the repeat function, if bit 7 is set, all keys repeat, if bit 6 is set to 1 and bit 7 to 0, the repeat is turned off altogether, and if bit 7 and bit 6 is off, only the above keys repeat. E.G.:- (fig 1c)

THE KEYBOARD

The front face of most keys now have two characters on, the left being accessed by the Commodore logo keys, and the right by either shift key. Pressing shift and logo swaps character sets. This is similar to the POKE59468, 12 or 14 on the PET. The control key gives access to colour control explained earlier as well as reverse mode. I think that this was left in the kernal from early on. The function keys give out new control characters which can therefore be sensed in a program. The 'Programmers Pack' will give these keys various functions such as RUN, GOSUB, LIST. I would expect that these words will be printed on the screen by a press of a function key.

When the restore key is pressed an NMI interrupt is caused. The NMI interrupt is used for RS-232 processing as well but that will be discussed later. The NMI servicing routine checks to see if it was the restore key that caused the interrupt. If it was, the routine checks to see if the VIC has a ROM pack present. If so, a jump indirect \$A002 is made. If no ROM pack is present, it checks to see if a VIC keyboard reset has been done (pressing RUN/STOP and RESTORE). It does a JSR UDTIM (this updates TI and sets the stop key flag) followed by a JSR STOP which returns .Z (zero flag) set if a stop key is detected. If a stop key is detected, the VIC chip registers set up as at power on, the indirect KERNAL vectors in page 3 are reset as well as the

(fig 1a)

```
CLR HME
10 REM A TEST
20 REM
  CLR HME
LIST
```

(fig 1b)

```
0 REM THIS IS A STUPI
D TEST
20 REM
```

(fig 1c)

```
POKE650,128:
POKE650,127:
POKE650,0 :
```

```
REM ALL KEYS REPEAT
REM NO KEYS REPEAT
REM ONLY CERTAIN KEYS
REPEAT
```

system vectors (interrupts I/O etc.), and the screen editor is set up (clear screen). Finally a BASIC warm start is done by jumping indirect \$C002. If no stop key is detected, the routine jumps to the NMI exit code.

Everything else works much the same on this keyboard as on the PET. I have only noticed one oddity, that is when a shift RUN/STOP is keyed, to load the first program off tape, only the left shift will work. The right shift simply prints LOAD, BREAK and READY immediately. This is useful however because the INPUT statement, is now bomb proof i.e. RETURN only will not drop out of a program, and thus using the right shift and RUN/STOP you can escape, but not with the left.

GRAPHICS

The VIC has all of the PET graphics which can be printed in any of the 8 colours BLACK, WHITE, RED, CYAN, PURPLE, GREEN, BLUE, or YELLOW. Like the PET, characters can also be POKE'd to the screen but their colour has to be set by POKE'ing the colour nibble table as well.

The VIC chip has a register that points to the start address of the character generator table, i.e. where it should get the dot representations of characters that are displayed on the screen. In the PET, this table is in ROM and not addressable by the 6502. In the VIC it is addressable by the 6502. This means that by modifying the register in the VIC chip the table could be situated in RAM and thus modifiable.

Each character is made up of a matrix of 8X8 dots (or 8X16 but I'll

leave this for now). Since each dot can be represented by one bit, (lit=1, unlit=0), 8 bytes are required for each character pattern. Each character codes occupies a byte, and when the VIC displays characters, it takes the character code from the video RAM, multiplies this number by 8 since each dot pattern takes up 8 bytes, and adds this to the address where the character generator table starts. This resulting address points to the byte which contains the first row of the dot pattern of that particular character. The following 7 bytes contain the other 7 rows of the dot pattern.

Since we can tell the VIC chip that the character generator table is in RAM we can produce programmable characters, and what I call pseudo high resolution graphics.

There are POKEs shown in the Japanese manual to tell the VIC chip that the character generator table is situated at 5120 (\$1400, RAM) instead of 32768 (\$8000, ROM) at power on. I have tried but still fail to understand how these POKEs work. VIC chip register CR5 (\$9005 or 36839) controls the address of the start of the character generator table, as well as the start of the video RAM. Bits 4-7 of this register correspond to bits 10-13 of the address of the video RAM, ACCORDING TO THE VIC CHIP, (I have specified this last point because the address decoding by the 6560 (VIC chip) may be different from the 6502). And bit 7 of CR2 (\$9002 or 36866) corresponds to bit 9 of the video RAM start address. At power on, the address of the video RAM according to the 6502 is \$1E00 or 7680, See fig 3a

Figure 3a

BIT#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIT	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
HEX				1				E				0				0

Figure 3b

BIT#	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIT	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HEX				8				0				0				0

Contents of bit 7 of CR2 =1 which does correspond to bit 9 of the 6502 address.

Contents of bits 4-7 of CR5 =111 which does not correspond to bits 10-13 of the 6502 address??

At power on, the address of the character generator table according to the 6502 is \$8000 or 32768, See fig 3b

Contents of bits 0-3 of CR5 =000 which does correspond to bits 10-13 of the 6502 address ??

After looking at the circuit diagram with John Rees aid, it appears that when bit 15 of the 6502 address bus is true i.e the 6502 is trying to access memory above \$8000 or 32768, bit 13 of the 6560 address bus is automatically set true ??

If someone can explain this to me I would be most grateful.

Given that I could change the character generator table starting address to \$1400 or 5120, I discovered the following:-

What I would call Pseudo-Hi-Res graphics could be produced:-

Basically, because up to 256 different characters can be displayed on to the screen at the same time, and the fact that they are ALL programmable, you can use this to do Hi-Res plotting, provided that not more than 255 squares on the screen are filled with hi-res (you must reserve one for a space, for background). However, literally as I write this blurb it has just dawned on me that if 8X16 size character are selected, that provides enough squares (512) to make the whole screen hi-res (MAGIC!). Hi-Res limits the video to 2 colours of course foreground (pixel lit), and background (pixel unlit).'

"Hark" I hear you say, "only two colours for decent graphics?" well no, you can have up to 4, but with half the horizontal resolution; That is 88 across by 184 down, instead of 176 by 184. The 4th bit (bit 3) of each colour nibble decides whether each character square is displayed in hi-resolution graphics (=0) or MULTICOLOUR mode (=1).

Hi-res gives a direct match of the bits set in the character pattern to the pixels colour

00011000		EEFFFBFB
00100100		EFBFBFBF
01000010		FFBFBFBF
01111110	= 'A' =	FFFFFBFB
01000010		FBFBFBFB
01000010		EFBFBFBF
01000010		FBFBFBFB
00000000		EEBFBFBF

0 - Background colour (B)

1 - The colour given by the colour nibble associated with the square - The foreground colour (F)

Multicolour gives a direct match to each pair of bits to a 'pixel' colour (each 'pixel' is two pixels wide)

00011000		EEFFFBFB
00100100		EFBFBFBF
01000010		EEBFBFBF
01111110	= 'A' =	EEAAAAFF
01000010		EEBFBFBF
01000010		EEBFBFBF
01000010		EEBFBFBF
00000000		EEBFBFBF

00 - Background colour (B)

01 - Exterior border colour (E)

10 - The colour given by the colour associated with the square - The foreground colour (F)

11 - Auxiliary colour (A)

Basically, all that happens in multicolour mode is that the horizontal resolution is halved, i.e the smallest controllable location is two pixels wide. This means that each location is now two bits wide and can thus take on the value zero through three which gives four colours per square.

Since there is a flag to decide the bit interpretation on each square, hi-res, and multicolour characters can be mixed. This would be good for labelling coloured graphs for example.

The auxiliary colour is controlled by bits 4-7 on CRE (\$900E or 36878). The numbers for the different colours are as follows and are the same for all colour control (border, foreground, background and auxiliary).

- 0 - BLACK
- 1 - WHITE
- 2 - RED
- 3 - CYAN
- 4 - MAGENTA
- 5 - GREEN
- 6 - BLUE
- 7 - YELLOW

Available for the background are 8 other colours because this has four bit control:-

- 8 - ORANGE
- 9 - LIGHT ORANGE
- A - PINK
- B - LIGHT CYAN
- C - LIGHT MAGENTA
- D - LIGHT GREEN
- E - LIGHT BLUE
- F - LIGHT YELLOW

As mentioned earlier, the 4th bit (bit 3) of CRF (\$900F or 36879), controls

invert mode (this only functions in hi-res mode). If this bit = 0, the VIC will display characters in the background colour, while the background assumes the colour of the square.

Serial Stuff

The VIC has a 'serial bus' and can drive the parallel user port like an RS-232 interface. As in the PET, device 1 is tape, 3 is screen, and 0 is keyboard. Since there can only be one cassette deck, device 2 is now RS-232 handling rather than the second cassette deck. Any other specified device will be handled by the serial bus. The serial bus has only ATN (attention), SRQ (service request), data in/out and clock in/out pins.

The serial handshake is buffered, i.e. on continuous transmission there would always be a pending character. (fig 4)

Graphics on the VIC-20

By Paul Higginbottom

I have come to the conclusion that implementing hi-resolution graphics on the VIC is all a matter of "slight of hand". I say this because when I read the specification of the VIC before I actually saw one, I imagined hi-res like a PET with a graphics board, four voice sound like a PET with a sound board. Well, unfortunately it wasn't that straightforward.

Note: The VIC chip registers begin at 36864 (\$9000), and are named CRO, CR1, CR2....CRE, CRF.

As an example of what I mean, I read that a pointer in the VIC chip (6560) would allow you to tell the VIC where the screen RAM starts and where the character matrices (character dot patterns) were addressed. It's not quite so easy I now know that this pointer only allows access to bits 10 to 13 of one address and bits 9 to 13 on the other and that these bits, are in bits 4-7 of CR5 (\$9005), and bits 0-3 of CR5 and bit 7 of CR2 respectively!

However, I still haven't figured out how these values work, and so I have produced a small program that will give you the values when given a screen address and a character set address (fig 5)

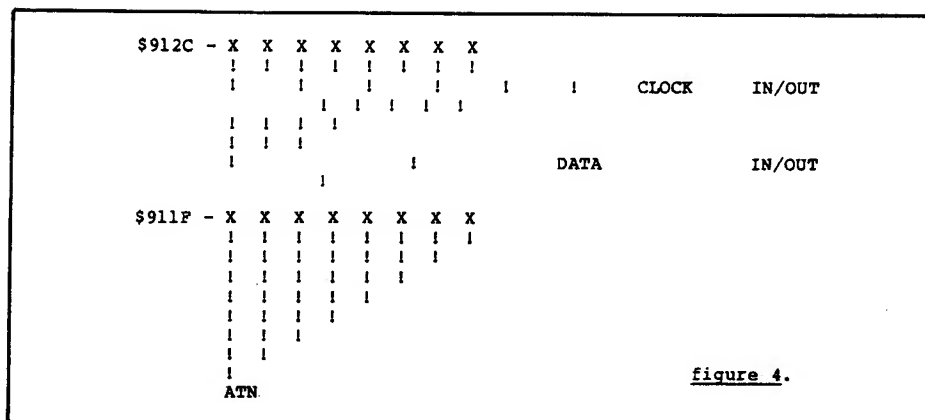


figure 4.

I wrote this program after looking through some source code of the VSP (Video Support Package). The screen usually starts at 7680 (\$1E00) and the character set usually starts at 32768 (\$8000). This program works with these values and some others I have tried. I am pretty sure that the minimum movement of where the screen starts is 1/2K and 1K for the character set. This is because bit 9 is the lowest accessible bit of the screen RAM start address, and bit 10 the lowest for the character set.

I have explained this because being able to change where the VIC reads its character representations from is the key to hi-resolution graphics. But I hear you say (as I did at first) "If there are only 256 characters, and there are 506 squares on the screen, doesn't that mean I can't use the whole screen for hi-res ??". Well, you can, because a 16X8 character matrix size can be selected. This means that, effectively each character occupies 2 squares (one above the other) on the screen and thus 256*2=512 squares are now available for hi-res (which is more characters than the screen occupies). Bit 0 of CR3 is used to select 16X8 character size (=1) or 8X8 (=0). Thus:-

POKE 36867,PEEK(36867) OR 1, will select 16X8 characters, and
POKE 36867,PEEK(36867) AND

254 will select 8X8.

So, now we all know how to tell the VIC that the character set is in RAM, and we can also tell it that each character is 16X8, i.e. each character requires 16 bytes of information from the character table. If you select the 16X8 mode on a VIC just after power up you will see some interesting effects. Since twice as much memory is required for each character representation, then pressing '@' symbol will give an '@' with an 'A' below it. This is because screen character code 0 is an '@' symbol and screen code 1 is an 'A' and when @ is pressed, a zero character code is put onto the screen, and the VIC takes 16 bytes of information from where character code 0 starts in the table, and this contains 8 bytes for the @, and then the 8 bytes for the A.

You will also notice that the bottom of the border has gone. That is because the screen is still set up for 23 rows and this is interpreted by the VIC as 23 rows of 16X8 characters and thus the screen now extends off the bottom of the screen and under the table you are working on!

Once 16X8 characters have been selected, and the character matrix table has been redirected to RAM, we can now implement hi-res graphics. If we clear the character matrix table first (by putting zeroes all the way

Figure 5

```
100 INPUT "SCREEN ADDRESS";SA:HB=INT(SA/256)
110 INPUT "CHRSET ADDRESS";CA:CB=INT(CA/256)
120 VD=((HB AND 253)*4) OR 128
130 AC=INT(CB/4) AND 15
140 IF AC<>0 THEN AC=AC OR 8
150 AC=AC OR VD:PRINT "POKE 36869,";AC
160 IF (HB AND 2)<>0 THEN 190
170 PRINT "POKE 36866, PEEK(36866) AND 127";
180 PRINT ":REM RESET BIT 7 OF CR2":END
190 PRINT "POKE 36866, PEEK(36866) OR 128";
200 PRINT ":REM SET BIT 7 OF CR2":END
```

Figure 6

Graphics on the VIC-20 by Paul Higginbottom

(This is in source code)

```
;SETUPS:-
;
*=$9000 ;VIC CHIP STARTS HERE

CRO **+=1
CRI **+=1
.
.
.
CRE **+=1
CRF **+=1
;
OLDTBL =$8000
NEWTBL =$1000
SCREEN =$1E00
;
PTR =$01
;
STEP 1.
;
LDA #252 ;CHANGE CHARACTER MATRIX TABLE TO $1000
STA CR5 ;BY SETTING VIC REGISTER 5
;
STEP 2.
;
LDA CR3 ;SET 16X8 CHARACTERS
ORA #1 ;BY SETTING BIT 0 OF CR3
STA CR3
;
STEP 3.
;
LDA #<NEWTBL ;PTR=(NEWTBL)
STA PTR
LDA #>NEWTBL
STA PTR+1
LDY #0 ;ZEROISE OFFSET
;
CLRTBL STA (PTR),Y ;ZEROISE ONE BYTE
INY ;BUMP OFFSET
BNE CLRTBL ;IF BLOCK NOT DONE - GO BACK
INC PTR+1 ;NOW BUMP HI BYTE
LDX PTR+1 ;GET IT INTO .X
CPX #>SCREEN ;REACHED THE SCREEN ?
BNE CLRTBL ;NO - GO BACK
;
STEP 4.
;
PUTCHR TYA ;.A=SEQ CHAR (=Y (OFFSET))
STA SCREEN,Y ;PUT SEQ. CHAR ON SCREEN
INY ;BUMP OFFSET (AND CHAR)
BNE PUTCHR ;IF NOT DONE - GO BACK
RTS ;RETURN FOR NOW
```

characters sequentially through the screen. Now then, there are 16 bytes per character in the character matrix table and so we must now multiply the character position (screen code) by sixteen. Having done that we now should have a pointer, which is pointing to the start of block of 16 bytes that define the character.

Still with me? I hope so.

We now have to add on the offset into the character which is Y AND'ed with 15.

Perhaps the diagram below may explain this one.

```
01234567012345670123456701234567
0 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
1 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
2 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
3 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
4 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
5 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
6 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
7 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
8 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
9 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
10 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
11 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
12 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
13 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
14 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
15 @@@@@@@@@AAAAAABBBBBBBCCCCCCCC
0 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
1 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
2 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
3 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
4 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
5 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
6 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
7 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
8 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
9 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
10 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
11 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
12 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
13 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
14 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
15 WWWWWWWWWXXXXXXXXXXXXXXXXZZZZZZ
```

Imagine each character as one pixel. Each @ pixel represents one dot in the character with screen code 0, etc.

Since each character is represented in the character matrix table as 16 bytes (one for each row), and each bit showing whether the pixel is lit (foreground colour) or unlit (background colour), and having established the address of the first byte of the 16, we must now offset into the block by the remainder of Y/16. By adding Y AND 15, this is accomplished.

Having found the byte to modify, we must now work out which bit to change. The bit position (going 0-7 from left to right) is (X AND 7) by the same reasoning as before. I couldn't be bothered with shifting and rotating the byte and so I simply OR'ed the current contents of the byte with a table value.

Assuming the .Y=0, PTR is pointing to the correct byte, and .X=(X AND 7), then:-(fig 8)

through it), this will make all characters appear as blank. If a screen code of 0 is put at the first location on the screen, and also at some other point on the screen, and we change the character matrix table, the new transformed character will show up at both points on the screen. This is not acceptable. (We can't have a hi-res plotting routine that when asked to plot a dot, it plots it in two places.) So what I did is to put the screen codes sequentially from position 0 on the screen and thus when a character representation is changed, only one character square will change. To recap: (see fig 6)

Now onto plotting. This now becomes fairly simple, if you are following up to now. Basically, all we have to do to plot a hi-res dot (given

an X,Y pixel position) is to find the character matrix that has to be modified, and modify it! This IS as easy as it sounds because of the way the screen is arranged:-

If the screen is 22 columns then :-

Character position = INT(Y/16)*22+INT(X/8)

This formula should be fairly self explanatory. Each character is 16 dots long, so INT(Y/16) calculates the row. We must multiply this by the number of characters in each row (22), and add on the column position INT(X/8) (since each character is 8 dots wide.) to calculate the absolute character position

The character position value = the screen code of the character to be modified now because we put the

to unplot a dot, the bit must be reset. I did this by having a different table, and ANDing it this time (fig 9).

One last point, the procedure for multicolour mode graphics rather than hi-res is very similar, except that X has half the range, and (X AND 8) becomes (X AND 4), the tables will be different because the colour of a

multicolor mode pixel, can be one of four 00, 01, 10, or 11.

That's all for this one, please let me and all other Commodore companies know, if you discover or develop anything.

Regards,

Paul Higginbottom

Software Advisor, Commodore Canada.

JOYSTICK CONTROL ON THE VIC

By Paul Higginbottom

figure 8.

```

LDA (PTR) ,Y          ;GET EXISTING VALUE
ORA BITMSK,X          ;TURN ON BIT
STA (PTR) ,Y          ;AND PUT BACK NEW VALUE
RTS                   ;THAT'S IT!

;
BITMSK .BYT %10000000
       .BYT %01000000
       .BYT %00100000
       .BYT %00010000
       .BYT %00001000
       .BYT %00000100
       .BYT %00000010
       .BYT %00000001

```

figure 9.

```

LDA (PTR) ,Y          ;GET EXISTING VALUE
AND BITMSK,X          ;TURN ON BIT
STA (PTR) ,Y          ;AND PUT BACK NEW VALUE
RTS                   ;THAT'S IT!

;
BITMSK .BYT %01111111
       .BYT %10111111
       .BYT %11011111
       .BYT %11101111
       .BYT %11110111
       .BYT %11111011
       .BYT %11111101
       .BYT %11111110

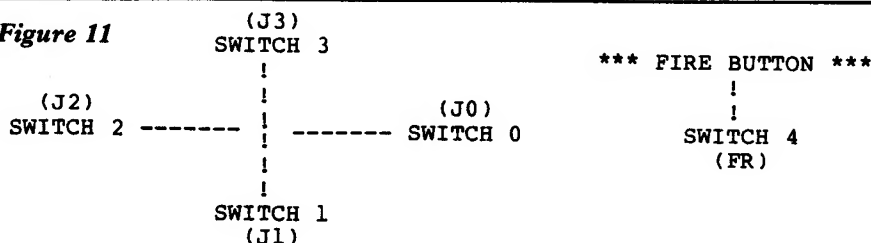
```

Figure 10

NAME	DESCRIPTION
D1DDRA	- PIA //1 DATA DIRECTION REGISTER FOR SIDE A
D1DDRB	- PIA //1 DATA DIRECTION REGISTER FOR SIDE B
D2DDRA	- PIA //2 DATA DIRECTION REGISTER FOR SIDE A
D2DDRB	- PIA //2 DATA DIRECTION REGISTER FOR SIDE B
D10RA	- PIA //1 OUTPUT REGISTER FOR SIDE A
D10RB	- PIA //1 OUTPUT REGISTER FOR SIDE B
D20RA	- PIA //2 OUTPUT REGISTER FOR SIDE A
D20RB	- PIA //2 OUTPUT REGISTER FOR SIDE B

These names are the ones used in the Kernal.

Figure 11



First, a note on the PIAs (peripheral interface adaptor) in the VIC-20 from a non-hardware oriented person!

The two PIAs in the VIC-20 we are concerned with are MPS 6522 chips. This chip is really two processors in one. This is because it has two independently processing sides; Side A, and side B. Each side has its own interrupts, clocks (these always count down, and can be set for timing by other processors) and I/O registers. I will name a few of these registers as in fig 10

The reason I have said this is so that the joystick reading algorithm may be understood. The convention I have used for the switches in a joystick are as follows:-(fig 11)

The BASIC variable names I have used are in parentheses.

Now then, the data direction register in peripheral number two, side B, is usually set up for read, and this output register has bits two to five connected to the joystick port in the following manner:-

SWITCH 1 = BIT 3 of D20RB

SWITCH 2 = BIT 4 of D20RB

SWITCH 3 = BIT 2 of D20RB

SWITCH 4 = BIT 5 of D20RB

(FIRE BUTTON

D20RB = \$911F = 37151 in the VIC and thus:-

J1 = - ((PEEK(37151)AND8)=0)

J2 = - ((PEEK(37151)AND16)=0)

J3 = - ((PEEK(37151)AND4)=0)

FR = - ((PEEK(37151)AND32=0)

(FIRE BUTTON)

These statements give the value 1 if the switch is pressed, and 0 if it is not pressed. Diagonal movements of the joystick will close two switches. The logical test has been made if the bit is zero. This is because the lines are high usually, and the closing of the switch grounds the line. The Negation of the statement is to make the value 0 or 1, rather than 0 or -1.

Now this is all very fine if you don't want to read SWITCH 0, but in most cases, you will need this one as well. This is where the snag is. This switch is connected to bit 7 of the output

register on peripheral number one, side A. This would seem to be fine, until we discover that this is not set up for read usually, because this register is also used for the keyboard scan. So, we have to set it up for read, read the register, and then set it back up for write, so that the keyboard continues to work. We only need to see bit 7 or D1ORA, so we put 127 in D1DDRA so that bits 0 to 6 are write, and bit 7 is read.

D1ORA = \$9120 = 37152, and...
D1DDRA = \$9122 = 37154 in the VIC. Thus:-

```
POKE 37154,127:REM ENABLE
D1ORA BIT 7 FOR READ JO
= -((PEEK(37152)AND128)=0-
):REM READ THE SWITCH
POKE 37154,255:REM RESET
ALL BITS FOR WRITE
```

Therefore, the final algorithm looks like:-

```
9000 REM READ THE
9010 REM JOYSTICK
9020 DD=37154
9030 P1=37151
9040 P2=37152
9050 POKE DD, 127
9060 P=PEEK(P2)AND128
9070 JO=-(P=0)
```

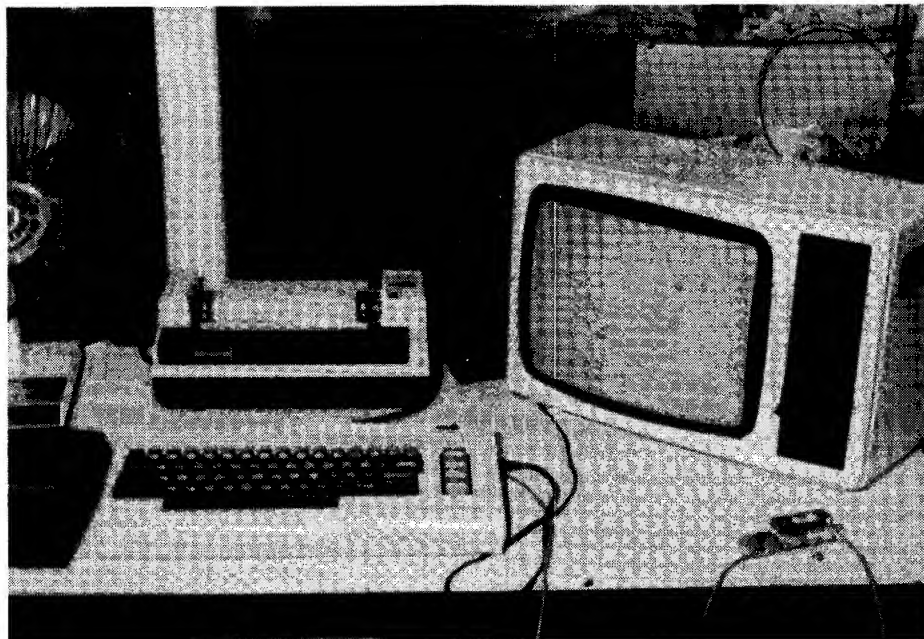
```
9080 POKE DD,255
9090 P=PEEK(P1)
9100 J1=-((PAND8)=0)
9110 J2=-((PAND16)=0)
9120 J3=-((PAND4)=0)
9130 FR=-((PAND32)=0)
9140 RETURN
```

An enhancement to this algorithm that I am now working on, is to have the routine return one value from

zero to seven. This would be the joystick position as in the diagram. And also a value for the firing button.

```

      0
      7 ! 1      F=button
6-----2  =J
      5 ! 3
      4
```



The VIC revealed

Workshop List Amendments

A number of amendments to the list as published in volume three issue 6. A point that would perhaps be worth bearing in mind, if you are considering applying to join the scheme, is an example set by a number of schools in the Dorset area. Rather than each individual school joining the scheme, with the expense incurred by copying of the tapes etc., and the time delays that a small establishment might suffer from, everything in that area is now handled by the local education authority. This has a number of advantages, not least of which is the pooling of resources into a central area, and the speed with which information now gets distributed.

ADDITIONS :-

Mr . M. D. Meredith Dept. of Education University of Southampton, Southampton SO9 5NH
MR. J. R. de Boer, Dept. of Civil Engineering N.E. London Polytechnic, Forest Road, London E17 4JB
P. Thompson (Schoolsoft), 67 Anderby Drive, Grimsby, South Humberside
Mr. A. Wood, 2 St. James Road, Dudley, West Midlands
A. Clark, Computer Studies Unit, Chichester College of Technology, Westgate Fields, Chichester, W. Sussex
J. Sansom, Computer Centre, Brighton Poly, Moulsecomb Brighton
Computer Unit, Bradford College, Old Building, Great Horton Road, Bradford BD7 1AY
C.R. Dean, Parkview School, West Avenue, Barrow-in-Furness, Cumbria /

No. of PETs	Courses for Teachers
4	Yes
15	Yes
3	Yes
6	Yes
12	Yes
50+	Yes
7	Yes
5	H & A

AMENDMENTS

Main contact at the London Borough of Haringey is now Mr. A. Lenney, Chief Education Officer.
Main contact at the Anfield Comprehensive School, Breckside Park, Liverpool, is now Mr. R. E. Marsh.
Main contact at the Computer Division, Bradford Council, Bradford, West Yorkshire, is now Mr. K. R. Tomlinson.
There is a change of address for one of the Dorset Centres. Instead of being :-
Mr. M. Finney, Blandford Upper School, Blandford, Dorset, to:-
Mr. D. Bale, Dorset Institute of Higher Education, Wallisdown Road, Poole, Dorset BH12 5BB (5 PETs, and they do courses as well).

and food and drink were available for us outside. Marvelous, we said, and indeed it was. There is nothing quite like coffee and hot bacon rolls at eight o'clock in the morning. Simply superb.

Suitably fortified we returned to the PET and carried on with the grand program, which in the end ran to about 4k altogether, not bad for a few hours work typing in 'blind', and going purely by a bit of script that we had.

Then onto the set, and praying that the PET wouldn't suffer from stage fright and die on us. 'Break a leg' we said to it, and duly satisfied it performed admirably. Having got it all up and running it was then simply a question of staying around to make sure nothing disastrous happened, and with that opportunity of seeing the B.B.C. at work, an interesting little process.

There seemed to be an awful lot of them for what they were doing, but knowing very little about the inner

workings of T.V. life they probably all had their allotted roles to fill. Certainly they all seemed to be involved a lot in what was going on, which was more than could be said for me and Chris. We were stoutly resisting the temptation to "do things" whenever somebody yelled "Action!", which they did at frequent intervals between lots of coffee breaks (not that many really) and going down to the pub (Fullers as promised) at lunchtime -the high point of the day.

All went well throughout the day, and the PET figured quite prominently in a number of scenes, and all our fears of disasters vanished as time wore on. Although we didn't in the end, feature in any of the scenes ourselves, despite earlier hopes of being discovered as the true stars that we are, the B.B.C. seemed quite satisfied with it all. I think the main impression we left them with was how easy it is to modify programs on the PET, one of its truly outstanding features.

So we left the happy beeb to their editing, put the PET to bed, and headed for the nearest public house to renew our acquaintance with Fullers London Pride. Anytime you need us, B.B.C., we'll be ready!

Day Trip to Paris

continued from page 17.

to access data. If the bus is active then Multex goes through the process again. If the bus is free then Multex takes control and passes the information it needs.

Multex is a very cheap way of allowing multiple PETs to talk to the same peripherals.

Both these products are available from Audiogenic (see back cover for address and 'phone number), and the prices are £69.50 for Multex, £39.50 for the Basic 2.0 Edex and £49.50 for the 80 column one.

PETALECT. An all-round computer service.

PETALECT COMPUTERS of Woking, Surrey have the experience and expert capability in all aspects of today's micro-computer and word processor systems to provide users, first time or otherwise, with the Service and After Sales support they need.

COMPUTER REPAIRS AND SERVICE

If you're located within 50 miles of Surrey, PETALECT can offer FAST, RELIABLE Servicing with their own team of highly qualified engineers.

24 hour maintenance contracts available. Our service contracts start at around only 10% of your hardware cost per annum for on-site, or if you bring it to us at our own service dept., it costs only £25 plus parts. Representing real value for money.

MICRO COMPUTER SUPPLIES

PETALECT can supply the great majority of essential microcomputer-related products promptly and at really competitive prices. Such items as:-

TAPES●PAPER●FLOPPY DISKS●PROGRAMMES FOR BUSINESS●SCIENTIFIC OR RECREATIONAL APPLICATIONS●MANUALS●COMPUTER TABLES●DUST COVERS RIBBONS●TOOL KITS●PRINTERS●ELECTRONIC INTERFACES WHICH ARE PETALECT'S SPECIALITY.

If you want to find out more about what we can and would like to do for you, why not give us a ring on Woking 69032/21776.

SHOWROOM

32, Chertsey Road, Woking, Surrey

We're worth getting in touch with.

PETALECT
COMPUTERS

SERVICE DEPT.

33/35 Portugal Road, Woking, Surrey

COMMAND-O: from Skyles Electric Works: distributed in UK by Calco @ £50

This chip gives the Basic 4.0 user many editing enhancements of Disco-pro, and provides (in some cases enhanced) versions of the well-known debugging and programming aids available in Toolkit, Basic Aid, Programmer's Friend etc., together with other interesting features.

This product is a 4k rom, which plugs into the UD 12 socket of the 8032, or UD 3 in a 4032. These are the rearmost socket and middle socket respectively. Those who suffer from socket congestion will require a Spacemaker, Rompager, or Socket-2-Me!

It is encouraging to note that the manufactureres have sufficient confidence in their product to go for the full ROM straight away, rather than choosing the less risky route of selling an EPROM until the bugs are all identified and eliminated!

It is worth noting the new trend towards the less-wasteful approach to making use of the spare sockets in the computer, by producing products using the full 4K of addressable space, rather than a mere 2K. Now that the prices of 4K EPROMS are falling, and EPROM programmers are becoming available, we can hope for this trend to continue.

The chip provides no less than 39 extra commands and can be fairly said to provide a great many of the capabilities which users would have liked to have seen incorporated in the original design of the operating system. Naturally there is room for improvement, and options will differ as to the relative merits of the various facilities, but it will be a very unusual user who will not feel that many of the facilities are a great help in programming, editing and debugging.

Some of the facilities are available singly elsewhere, at an individual price of one quarter of the price of Command-0, and on that basis the chip must represent very good value for money. Indeed, apart from Commodore's own Basic Aid for Basic 4 machines, (not yet available, but likely to be less than £10.00 when it is), it is difficult to think of any add-on

facility which offers so much for so little.

Many of the most frequently-used commands can be invoked by a single keystroke, followed by RETURN. This is most convenient, and the appropriate keys are easily learned, although where the instructions are similar to some of those familiar to users of "The Wedge", I would have preferred to see the same key used: i.e. "up arrow" for "load and run", rather than EXECUTE. Almost all the other commands can be called by two keystrokes, the second one being shifted.

One feature of some importance is that most of the new commands are fully tokenised, and can therefore be incorporated in programs. Some would not be of much use in this role, but others would. In addition, some of the commands can be used together in extremely powerful ways. In particular SET permits some exotic treatments, when used in conjunction with others. The manual suggests a short routine which will execute the last command you put on the screen, LIST and RUN the program, give HELP as to why and where the program stopped, and DUMP the variables. Finally this little gem will leave you the lines you were last entering, some three seconds before!

The manual is an enormous help, the combined work of Greg Yob and Jim Berkey. It is disarmingly frank about the "Gotchas", and contains a very large amount of information designed to help you get the best out of the product. It is also refreshingly critical in its approach, and in places reads much more like a fearless review of the Command-0, than a manual extolling its virtues!

Turning to individual commands:

EXECUTE is particularly useful in that it loads and runs a program from disk, without your needing to specify which drive it is on. (This is a feature of a number of commands, which saves your waiting for a file to be found, only to have the machine fail to find it because you failed to specify Drive 1, and the machine searches on-

ly Drive 0. In addition, "Universal Wedge" "hangs" up the machine in these circumstances!) SEND will send any string to the Disk unit, or if some other device is specified, to that device until a new device number is set. This enables easy communication with your printer, and is particularly useful if you have more than one disk unit connected to your printer.

MERGE enables you to append programs from disk onto the end of the existing program, and MERGE £ permits a program overlay; the manual gives very detailed instructions on how to avoid the destruction of variables when merging in new sections of program. INITIALISE will be attractive to users of DOS 1, since it initialises the disk drive selected, on the unit selected. MOVE puts the cursor at an absolute position, referenced by column and row. AUTO gives line numbering, DELETE does what you expect! FIND locates a string or BASIC keyword. It is enhanced in that the first matching character is placed in reverse field. Also a line containing the string or keyword more than once, is listed once for each time it occurs! Improvements here would have been output to a printer, and/or printing only a screenful at a time. More serious, however, is the fact that no CHANGE command is implemented. The ability to change a string name for another more-meaningful one, and to change the cursor control characters to something easily understood is very welcome, and some other products have this.

HELP shows where the error which caused a SYNTAX ERROR message to be printed occurred. TRACE is an improved version which prints the whole line at the top of the screen; if the line contains multiple statements, a reverse field marker shows the one being executed. The repeat key gives continuous operation, at a speed set by the user; any other key acts as the familiar STEP command. I would have welcomed the ability to print the TRACE to a printer, and to trace the values of selected variables. The fact that TRACE may be included in a program enables you to start and

finish the TRACE as part of the program which is giving trouble. (OFF disables TRACE, and is similarly programmable.)

A very powerful PRINT USING statement permits the formatted printing of numbers of strings, lining up decimal points, commas for thousands, currency symbols, supplying justification, and rounding off. The image string can be up to 79 characters long, and multiple image fields are permitted, up to a maximum length of 40 characters. The manual is particularly helpful in providing much detail about this facility. RUN runs the program in memory at the touch of the left arrow, which is very convenient, once you are used to it. BEEP allows the setting of the duration and tone of the bleeper. Rather useful for giving audible warnings in programs. EAT is a new facility set on the RUN/STOP key, whereby any characters to the right of the cursor are drawn leftwards to the cursor's location and erased! RENUMBER does everything that you could wish, in that it works at machine code speed, and renumbers by four parameters, so that you may

renumber a subroutine to start at a particular line number and increment by whatever amount you wish, knowing that all references will be adjusted. It is the users responsibility to see that the renumbered block does not clash with higher line numbers, but this is not difficult. Since the screen is used during renumbering, its limited capacity means that programs longer than 16k should be renumbered in two parts. Also the screen is blanked after the renumbering operation, which can be inconvenient.

DUMP provides a listing of variables, although not subscribed ones. (This mars an otherwise very attractive package. Many debugging operations require examination of the effects of loops on arrays, and yet only one package, Peter Dowson's PDAS, has ever tackled the problem.) Dumping to a printer, and in screenfuls would have been desirable.

SCROLL is a most remarkable development, and permits you to scroll up and down through a BASIC program as if viewing it through a window. This is not only fascinating, but a great time-saver, and likely to

avoid a lot of listing on a printer, since you can have a look at any part of a program at will.

SET is a new departure also, in that it allows the pressing of any key to print a string of up to 70 characters to the screen. The string may include chr\$(13), so that you can in fact write a small program on the screen and cause it to LOAD and RUN as soon as the defined key is touched. This has many possible applications, and when you consider that the set key could call up a named subroutine, or print a named string, which could be charged under program control, you can have an automatically self-modifying single key subroutine!!!

All the new 8032/8016 control characters are available at the touch of two keys, instead of the rather cumbersome method given by the operating system. In addition, any key held down while "escape" is pressed becomes a control key: e.g. "control g" = ASCII 7 = bell.

I feel that most serious programmers will want this product. Despite the criticisms above, I am convinced that the chip is well-designed and represents a substantial step forward.

LANDSOFT SUPERIOR PROGRAMS FOR THE CBM PET

PAYROLL PLUS

£150 + VAT

This must be the finest PLAIN PAPER PAYROLL system available for the CBM PET.

It is designed to the Inland Revenue Specifications for Computerised Payroll. The program is very 'user friendly' and should present no problems even to those who have had no previous computer experience. The manual is written in simple language and avoids computer jargon.

WORDFORM

£75 + VAT

This remarkable MACHINE CODE program will solve the problem of the majority of PET owners who desire high-grade word-processing capability but cannot really justify the usual high prices associated with the better packages. It will literally perform 90% of the functions of the expensive programs, and it would be rare to require the extra few functions in actual use.

See them at your approved dealer

Published by LANDSLER SOFTWARE 29a Tolworth Park Road, Surbiton, Surrey Tel: 399 2476

Continuing the theme of word processing, this month we will examine in depth, another major program, Wordcraft. This is a machine code word processor, working on either 40 column or 80 column Commodore machines, according to the user's selection at the time of purchase.

The package comes on an 8050 or 4040 type diskette, and the first action should be to take a copy of this, before using the program.

Security is provided by means of a plug-on chip, affectionately known as "the dongle". This device is plugged onto the cassette connector at the rear of the machine, and has set a new standard in protection devices: On the one hand, the program can only be used on a single machine at any one time, and the security device is either impossible to copy, or too much trouble for any potential pirate, and on the other hand, the user does not need to open the machine, risking bending pins on ROMs, does not use up one of the previous slots in the computer, and can carry the program from machine to machine if desired.

The Reference Guide which accompanies the program is over 80 pages long, and somewhat technical in nature, as befits a complex program. However the first-time user is likely to be somewhat daunted by the task of mastering such a manual and the author, Peter Dowson, has enlisted the aid of Mike Lake in a sensible approach to the problem: Eighteen sturdy A4 TRAINING CARDS accompany the program, so that the newcomer, instead of feeling adrift in a sea of complexities, is led at a gentle pace through the various stages of knowledge. This approach has much to commend it, and makes learning about the package a straightforward and pleasurable process.

The program formats the material as you go, so you can see how the text will look, unless you specify a width of document which is greater than the width of your screen. If you do this, the text spans across the screen in jumps, so that you can see a little beyond the point you have arrived at. Users of 40 column machines will find it convenient to format with a width of 40 columns so that they can see all the text as they go along, and to change the width to that which is required, after they have typed the text, so as to see most easily how the docu-

ment will look when printed.

The program is remarkably versatile in how it can be adjusted to match up to various printers, thus making best use of the facilities of each. Bi-directional printing and even white-space justification will be accomplished, if the printer permits this. Subscripts, superscripts, underlining and emboldening are all available. Pitch may be adjusted also, and special daisy-wheel characters are supported, including all of those on Qume and Diablo printers.

Unusually, Wordcraft will work with a large number of disk drives simultaneously, so that up to four units in all can be accommodated! This is very convenient in terms of the ability to pull in text files freely, and store them freely. By virtue of a little bit of smooth programming, it is not necessary to specify the device number, since the eight drives are given numbers 0 to 7 inclusive. The default is the last one specified. How pleasant it would have been if Basic 4.0 had adopted the sme protocol! This is just one example of the painstaking thought which has gone into this package.

The diskette-handling facilities in Wordcraft are extensive, and they need to be because of the subdivision of documents into chapters. All the usual housekeeping facilities for copying, backing up, formatting etc are covered, and the formatting of diskettes under Wordcraft automatically results in program overlays' being transferred to the newly-formatted disk. A special document directory may be printed to the screen or to the printer, and this shows the full details of the document concerned, including filename, date, diskette identity and name, number of chapters, pages in each, and length of each chapter, in characters. Some formatting information is also provided. The full disk directory can be brought to the screen also, although not printed out to the printer. The ability to copy a docu-

ment fileset in one operation is also provided, which is a welcome convenience.

Support is given for each of the Multiple Pet Systems on the market, and spooling can be carried out, to enable a Wordcraft file to be printed later by another computer. A recent development is that facilities are now incorporated so that any sequential file, whether in true ASCII or in Pet code, can be read into Wordcraft for editing and/or later printing and/or saving for future use. This is a major step, and increases the versatility of the program very much.

Direct links also already exist for Wordcraft to interface with Compsoft's Database Management System, and Petact now market a program to change Visicalc files so that they will be useable directly as inputs to Wordcraft, enabling them to be tidied up for proper presentation, including adding more informative headings and labels at the left of the tables produced by Visicalc. This is particularly advantageous, since there is a built-in difficulty in using Visicalc, in that all columns must be of the same width, within a window.

The latest version of Wordcraft, F 2 at the time of writing, gives space available for text of 11000 characters to be in the RAM at any one time. This is a relatively small amount of text, and the program relies heavily upon disk usage to avoid inconvenience from this. The 80-column version of the program gives a continuous display on the screen, the text is stored in a compact "packed" format, for efficient store usage. This has a cost in terms of processing time, such that delays of up to 3 or 4 seconds can occur when jumping about editing documents, especially towards the end of long documents. Whether this irritates you will depend on your Impatience Quotient, and also to what extent your eyes are glued to the screen, whilst typing!

To speed up processing, the document files are stored in screen code format, so that they need special handling if they are to be used by other programs. However, an appendix to the manual gives full details of how to handle this, even to the extent

of listing an Assembly Language program for reading blocks of bytes from an open disk file!

The screen layout is dominated by a five-line header, which is perhaps a little extravagant! However, the penalty for having the formatting codes imbedded in the text, so that the screen shows almost exactly the layout of the finished document (with the exception of right-justification, underlining, and certain visible control characters), is that you need rather more information available than would otherwise be the case. The facts given include: the name of the document, (not necessarily the filename!), date of the document, page number, and total number of pages in the document, the column and the number of columns used, the line number and the number of lines to the page, and finally, the mode in which you are at present. The 80-column version also gives details of the disk drive last accessed, and the chapter number.

There are no less than 30 COMMANDS available in COMMAND MODE, and 35 CONTROLS available in CONTROL MODE.

You toggle between CONTROL MODE and TYPE MODE (in which you enter text), by hitting RUN/STOP. To enter CONTROL MODE, you hit OFF/RVS.

Which MODE you are in is important, since hitting "n" when in COMMAND MODE will delete your entire text file from the computer's memory! This would be highly dangerous, but Wordcraft provides for the situation by asking for confirmation that you really want to do this, by giving a response "NOT SAVED! SPACE = DO COMMAND, STOP = DON'T". Very good practice, other software houses please copy!

Wordcraft is page-based, so that you can see exactly how each page will look. You can go to the top of the page by hitting HOME and to the bottom by hitting SHIFT/HOME.

The cursor keys allow you to move about within a page pretty quickly, but really impatient users may speed up editing by setting tab stops at convenient intervals across the screen, for really fast movement! To go to the next page on, you must hit RVS P CURSOR DOWN. Similarly it is possible to move through the pages, and to go to a page by specifying it by number.

Deletions are handled in a way

which will be sensible for typists but irritating at first for regular Pet users. The DELETE key deletes the character under the cursor, not the previous one! Words are deleted by RVS D, and sentences by RVS DEL. Insertion of single characters is handled by normal use of the INSERT key. Inserting a section of text is done by opening up a section of spaces by hitting RVS INSERT and beginning to type. Afterwards you hit SHIFT RVS (OFF), and the remaining spaces are deleted.

Margins and Tab stops are set using a "RULER". The default margins are, very sensible, for A4 paper, with 12 characters to the inch. Curiously enough, the program defaults to right justification, which can be easily enough cancelled, but opinions will differ as to whether this is the best idea. Much word processing is carried out on business letters, and many addressees are likely to be offended if they receive letters which they can identify as being "From a computer." One of the objects of using a daisy wheel printer is to ensure that the fact that a computer has been used is obscured from the recipient, and I feel that right justification gives the game away. Even the ability to eliminate typing errors may do the same, and some crafty businessmen may be tempted to use a little error-correcting fluid, to make it look as if the typist is less than perfect after all!!!

The settings of the margins and tabs are visible as you type, which is a substantial advantage.

Remembering that RVS is the CONTROL key in Wordcraft, then to set tabs you use RVS#, to clear you use LEFT ARROW. You use CONTROL RETURN for forcing new lines, and to start a new paragraph after leaving one or more lines, you use CONTROL + n, where "n" is the number of lines required.

Centering text is by CONTROL "=", following a forced line, and followed by the same. You force a new page by CONTROL HOME.

Text highlighting is set up by enclosing the relevant portions in brackets. There is a print-time option to select between boldening and underlining, to override the default option of underlining.

Control "w" sets a WAIT POINT for printing, which pauses the process until you have adjusted the paper

perhaps, or changed the daisy wheel.

The imbedding of CONTROLS within lines can lead to some confusion as to what might happen if that line were moved or edited out, accordingly Wordcraft offers the ability to examine them (CONTROL "c"), delete them selectively (CONTROL "n" "x" with "x" as the type of CONTROL to be deleted.) All controls on any line can be deleted by CONTROL "n" "c".

Indented paragraphs are achieved by using CONTROL and square brackets to enclose the text, having tabbed to the point required beforehand.

Blocks of text are identified by cursor placement, and can then be moved around, deleted, or reproduced by the use of commands which are virtually mnemonics, e.g. "m" for MOVE, "r" for REPRODUCE and so on.

Backspacing so as to overprint is achieved by ESC "!", which is useful for insertion of accents and to form special characters. If your printer does not have a £ sign, it may be possible to form one in this fashion.

Wordcraft uses a system of "Fill points" to permit letters' being prepared, and the gaps filled later, either manually, or automatically from a 'Fill file'. The program takes you automatically from Fill point to Fill point by means of CONTROL "?"

SEARCH is a powerful function, enabling you to find any string of up to 30 characters, either once or each successive occurrence. A "?" acts as a wild card, so that it may be included in the string to catch a variety of versions of the string, for example, including capitals or lower case letters. SEARCH AND EXCHANGE, either once or for every occurrence is also available, very helpful in a labour-saving way in enabling you to correct a reference, or to use abbreviations of common phrases, so as to replace them in a single operation at the end of preparing the document. The limitations of the function is that it is available only so far as the current chapter is concerned, although I understand that the possibility of its operating globally is being examined currently. This is of particular importance to people who produce long reports, and book manuscripts.

Standard documents can be prepared easily from a file of standard paragraphs. Each paragraph is treated

as a separate page, the pages being called up by numbers.

Printing options are extensive, you may print page by page, a selection of pages by number, or the entire chapter of pages currently in memory. Multiple copies can be printed, and double spacing is possible, although treble spacing is not. It is possible to vary the number of lines to the inch on those printers which support this, and pitch (characters to the inch) may also be specified, 10, 12, or 15 characters to the inch being available.

A standard page length of 66 lines is assumed, and Wordcraft gives you 55 lines on that sheet unless you specify something different. Page numbering may be provided automatically, and you may start at a number other than one required. Both left and right HEADERS and TRAILERS are available, as is continuous or single sheet printing.

The screen scrolls whilst printing is taking place, so you can see where you have reached. Global printing of all chapters of a specified document, in one operation, without user intervention is available.

The form of **BACKGROUND PRINTING** provided is a large step forward. The disk unit controls the printing of a special file previously saved onto a diskette, so that editing or typing in of new text may continue. This frees up the computer, instead of its being tied up during slow printing operations. During this process, no editing of the document being printed may be carried out, but the system has been very well thought out. The manual is engagingly frank about the degree to which this system can be relied upon, because some printer interfaces are not sufficiently sophisticated to allow Wordcraft to interrupt the flow of data to the printer. (Indeed, some interfaces will not carry out background printing at all! This includes my interface, and I have been unable to test the background printing myself at a result!. Potential buyers are warned to make extensive enquiries, or better still, to see the **BACKGROUND PRINTING** in operation!). The system is somewhat experimental. However, it has proved to be very reliable in trials, and the manual goes to some trouble to tell you how to restrict its use if necessary, and how to escape from error conditions is encountered. The beauty of the system

is that the flow of data to the printer is stopped for a message to appear on the screen indicating that the printer needs some attention (e.g. Change a sheet of paper.), whilst background printing is in process. As soon as this is attended to, you can carry on typing as usual. Furthermore, and **this is quite remarkable**, the disk unit may be accessed for saving or calling up files, the only result being that background printing will pause briefly. In some instances, the existence of this versatile background printing facility, could enable users to avoid the need to buy extra hardware, in order to maintain production.

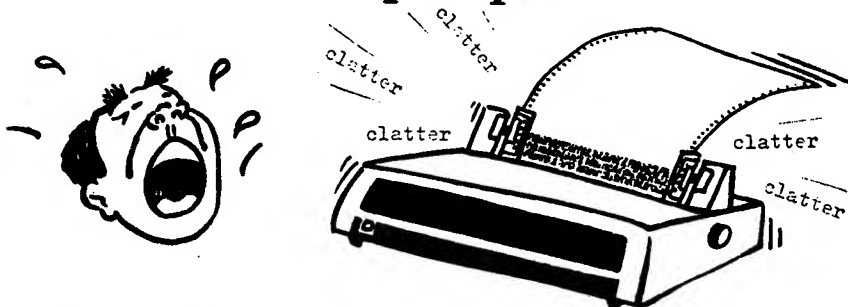
The program diskette contains a number of examples, to help you master the complex range of options available.

Various protections are provided against user error. If an attempt is made to clear the memory of a docu-

ment not yet saved, you will be warned before continuing. Also you cannot load up a new file without saving the one in memory, if this has not previously been saved. Similarly, any disk error causes the program to print an error message, and you must hit SPACE to continue. This is one of the best features of Wordcraft, since it makes sure you don't fail to know that something is adrift. I would favour the use of the (unfortunately quiet) bell as well, so that touch typists do not happily type away in ignorance!

All in all this package maintains its reputation as one of the leading factors in making the PET computer capable of being taken seriously in a business environment, and once its complexities are mastered, it will enable the computer to make a cost-effective contribution to the efficiency of document preparation.

Keep it quiet.



FEEL LIKE SCREAMING!!!!?BUT CAN'T HEAR YOURSELF SPEAK FOR PRINTER NOISE!!!!?

ALL YOU'RE LOOKING FOR IS THE STRONG QUIET TYPE!!!!*NOW AVAILABLE!!!!*!!!!*

THE 'PRINTERPROOF' ACOUSTIC BOX

SCIENTIFICALLY DESIGNED TO IMPROVE YOUR WORK ENVIRONMENT

- * HIGH SPECIFICATION
- * LOW PRICE
- * EASILY INSTALLED
- * RESTORES SANITY IN SECONDS

What they Say:

"I am very impressed." John Mundy, Bestobell Acoustics.
"A convincing 14/15dBA reduction achieved on test." Lab.data.
"Are you sure the printer's working!?" End User (delightedly).
"We can actually hear ourselves on the 'phone!!" User's staff.

* * *

This fully-tested "anti-clatter" unit reduces operational noise-level to General Office criteria - ISO Rating 55. Precision-made and built to last in sound-deadening heavy-gauge aluminium with tough Perspex viewer and foam-lined for maximum sound absorption, the

'PRINTERPROOF'

completely encloses the printer yet gives easy access for switch-functions, paper-changing and maintenance and is attractively texture-finished in Black and Cream. None of your plastic collapsibles with the built-in rattle - this is the right one!!

Internal Dimensions: 18"W x 15"D x 8"H. (Fits most CBM style printers).
Price: £79 + £2 carriage + VAT buys the best acoustic cover on the market yet we maintain our policy of a full money-back guarantee if you are not satisfied and return it to us within 14 days.

Your cheque with order, please, to
'PRINTERPROOF', 12 Wokingham Road, Reading RG6 1JG
or telephone Peter Stayne on Reading (0734) 661432 for any information.

'PRINTERPROOF' IT FOR YOURSELF!

Basic Programming

DIMP AND THE INTERRUPT

```
0 GOTO0001
1 PRINT:PRINT"HELLO THERE M":POKE1030,51:SYS826
5 PRINT:PRINT"NOW DO A DIMP":GOTO3005
3001 PRINT:PRINT"TRY AGAIN "
3005 SYS820:PRINT:PRINT:PRINT:GOTO5
```

```
826 33A 78      SEI
827 33B A9 50    LDA    ##50
829 33D 85 90    STA    $90
831 33F A9 03    LDA    ##03
833 341 85 91    STA    $91
835 343 58      CLI
836 344 60      RTS
837 345 78      SEI
838 346 A9 2E    LDA    ##2E
840 348 85 90    STA    $90
842 34A A9 E6    LDA    ##E6
844 34C 85 91    STA    $91
846 34E 58      CLI
847 34F 60      RTS
848 350 A5 09    LDA    $09
850 352 C9 01    CMP    ##01
852 354 F0 07    BEQ    $35D
854 356 A5 3B    LDA    $3B
856 358 F0 03    BEQ    $35D
858 35A 4C 2E E6 JMP    $E62E
861 35D A9 00    LDA    ##00
863 35F 85 09    STA    $09
865 361 A9 01    LDA    ##01
867 363 85 3B    STA    $3B
```

```
869 365 78      SEI
870 366 20 A7 C5 JSR    $C5A7
873 369 20 A4 C7 JSR    $C7A4
876 36C 4C 89 C3 JMP    $C389
879 36F 60      RTS
880 370 A5 77    LDA    $77
882 372 48      PHA
883 373 A5 78    LDA    $78
885 375 48      PHA
886 376 20 6F C4 JSR    $C46F
889 379 86 77    STX    $77
891 37B 84 78    STY    $78
893 37D 20 70 00 JSR    $0070
896 380 20 95 C4 JSR    $C495
899 383 20 93 03 JSR    $0393
902 386 20 70 00 JSR    $0070
905 389 20 00 C7 JSR    $C700
908 38C 68      PLA
909 38D 85 78    STA    $78
911 38F 68      PLA
912 390 85 77    STA    $77
914 392 60      RTS
915 393 A9 23    LDA    ##23
917 395 85 01    STA    $01
919 397 A9 D1    LDA    ##D1
921 399 85 02    STA    $02
923 39B A2 50    LDX    ##50
925 39D A0 0C    LDY    ##0C
927 39F BD FF 01 LDA    $01FF,X
930 3A2 D9 CA 03 CMP    $03CA,Y
933 3A5 F0 07    BEQ    $3AE
935 3A7 CA      DEX
936 3A8 D0 F5    BNE    $39F
938 3AA 88      DEY
```

Dimp & the Interrupt

An obvious problem with DIMP (CPUCN, Vol 3 issue 2 1980) is the inability to continue in program mode in the event of an unacceptable entry. Thus a syntax error or an overflow error causes the real immediate mode to be reinstated, and RUN must be entered to reinstate the dimp mode. Additionally, LIST will reveal all the secrets, including those of DIMP.

In the attached example, the interrupt is moved to \$ 350 where it is used to test for the real immediate mode and the LIST command. If either of these is found, the interrupt clears the problem and performs the Basic instruction GOTO 0, thus running the DIMP program again. Line 0 is therefore essential to the program, as it is required in the event of a dimp error. The error line is easily updated as required, and in the example POKE 1030, 51 rewrites line 0 to read GOTO 3001. The error line prints "TRY AGAIN" and resumes the dimp, without the loss of any previously endimped information.

In the event of a closed loop being desired, the interrupt final jump to \$E62E may be altered to \$E631 in byte \$ 35B, thus disabling the stop key in the usual way.

The table of bytes commencing at \$3CB consists of those tokens which have been designated as illegal quantities for this dimp. In the example they are : LOAD,SYS,STOP,END,-,SAVE, INPUT, NEW, INPUT , OPEN,CLOSE,VERIFY,POKE. The number of bytes in the table is entered in byte \$39E.

```
939 3AB D0 F2    BNE    $39F
941 3AD 60      RTS
942 3AE A9 20    LDA    ##20
944 3B0 A2 50    LDX    ##50
946 3B2 9D FF 01 STA    $01FF,X
949 3B5 CA      DEX
950 3B6 D0 FA    BNE    $3B2
952 3B8 4C 23 D1 JMP    $D123
```

TABLE OF BYTES STARTING \$3CB : 93, 9E, 90, 80, 94, 85, A2, 84, 9F, A0, 95, 97

Eliminate

1. Errata. There is a mistake on the 4040 Demo Disk. Lines 40,41,42 and 120 of the program "Copy Disk Files" contain the variables DS and DS\$ (for Directory Size).

These variables cannot be allocated with a Basic 4 System, since they are reserved for Disk Errors. I simply changed them to D9 and D9\$ - which took care of the problem.

2. I had the pleasure to glance through your "Best of CPUCN" and a little program called "Eliminate" caught my attention.

I very often have to delete parts of programs - sometimes I start a new program by loading an old one and deleting the unnecessary routines. This is a cumbersome procedure and I was happy to take "Eliminate" and fiddle around with it a bit to convert it into a regular utility program.

Here is the result. This program can easily be appended to any other program by using the screen editor. After typing RUN 60000 it will prompt for starting line, line increment and line interval. Since it deletes blocks of 8 lines each, it might stop before deleting the last line(s) - but it will tell you to which line it deleted.

Naturally, the program can be self-eliminated by "RUN 60000" "60000" "10" "60070"

From P.Gabar

Adding Records

```

100 OPEN 1,8,3,"0:ADRESS,SEQ,READ"
110 OPEN 2,8,4,"0:NEWFILE,SEQ,WRITE"
120 INPUT 1,A$
130 IF A$ + "ZZZ" THEN 200:rem TEST END OF FILE
140 PRINT#2, A$;CHR$(13);
150 GOTO 120
200 INPUT"SUPPLEMENTARY RECORD";A$
210 IF A$ = "END" THEN 300: REM TEST END OF INPUT
220 PRINT#2,A$;CHR$(13);
230 GOTO 200
300 REM END OF PROCEDURE
310 PRINT 2, "ZZZ";CHR$(13); : REM PRINT END OF FILE
320 CLOSE 1 : CLOSE 2
400 OPEN 15,8,15
410 PRINT#15, "0:ADRESS" :REM SCRATCH
420 PRINT#15,"0:ADRESS=0,NEWFILE" : REM RENAME
430 CLOSE 15:END

```

Adding Records

If you have a sequential disk file where you wish to add some records, there is a system which permits you to gaid a few instructions, and time. But first let us consider the classical version of thew problem:

f.i.: An address file (named "ADDRESS") where we would like to add a supplementary record through the keyboard (using INPUT). As this file is of unkown length we had previously put an End Of File flag ("ZZZ") as last record.

The succession of operations will be thus :

- 1° To record on a new file (named "NEWFILE") the content of the address file but **not** the end of file flag.
- 2° To input the new records and record them on NEWFILE
- 3° To record the end of file flag (ZZZ) on NEWFILE
- 4° To scratch Address file
- 5° To rename the file "NEWFILE" in ADDRESS"

A typical sequence of instructions will be as the diagram above.

By doing so there is no need for scratching and renaming

It seems that the replacement and the disk directory will be updated only when the files are closed.

from M.Shongut.

ELIMINATE

READY.

```

60000 INPUT"STARTING LINE NUMBER";S$
60010 INPUT"LINE INCREMENT";I
60020 INPUT"LAST LINE";L:S=VAL(S$)
60030 PRINT"OK" :FOR K=1TO8:PRINTS+(K-1)*I:POKE622+K,13:NEXT
60040 PRINT"S="S+8*I":I="I":L="L":S$=STR$(VAL(S$))+"" :POKE632,13:POKE631,13
60050 PRINT"GOTO60050" :POKE158,10:END
60060 IF S<L-7*ITHEN60030
60070 PRINT"OK LINES #";MID$(S$,2);" TO #";MID$(STR$(S-I),2);" DELETED!":END
READY.

```

Where to Start

Last time we discussed the possibilities of upgrading your systems, and how far you can go with any particular piece of hardware. Now we move onto considering software, and the type of thing that is available for whatever particular requirement you have in mind.

The amount of software available for the PET is one of the major reasons for its enormous success. But this range of software is also perhaps bewildering for the newcomer to the scene who, quite reasonably, wants to make sure that what he's buying is capable of doing the job, before spending the money. Your local dealer here can be a valuable asset, and it is well worth cultivating a good relationship with that dealer whilst you're considering what is available. Still, it's useful to have some guidelines before you start.

Possibly the most commonly used applications on the PET are the Word Processing, Stock Control, Payroll and Data Base management. Now a lot of programs are being written that combine these processes, or at least link into other programs that do one or other of the functions. For instance, DMS from Compssoft is a very good data base program, with links to Visicalc for financial forecasting, and Wordpro or Wordcraft for word processing. But we're running ahead of ourselves, so let's get back to basics.

In the last issue of the newsletter, you'll have read the reviews of two of the products mentioned above, namely Wordpro Four Plus and Visicalc. Another one is featured this month, and that is Wordcraft. DMS we featured a while ago now, so by looking at those you can get some idea of what those powerful programs offer. Since we've looked at those four earlier then, let's take a look around at some of the other programs available.

If I spent space on describing each program in great detail, not only would I be repeating what has gone before but also taking up the next ten million newsletters as well, so I simply present the programs and the 'phone numbers and leave the rest to you.

Payroll first of all. There are a number of packages around, from

Computastore (061 832 4761), Mitre Finch (0904 52995), Intex Datalog (0642 781193), Landsoft (01 399 2476/7) and of course Commodore themselves. As far as I'm aware, no-one has yet linked any of these payroll packages into anything like Wordpro, Visicalc etc., but no doubt it will happen soon. If you're in the market for a payroll system it's worth ringing all of them up in turn, because although payroll is a program that you can't really get wrong (after all, it's all about facts and figures that have to be right by law, as opposed to something like a wordprocessing system that is very much up to the writer's imagination), for every company that will take and use an "off-the-shelf" package there is one who require a package that might well have to be slightly tailored for their own requirements.

Tailoring in this fashion is something that Commodore cannot do - with 50,000 PETs out where we'd have rather a hard time looking after you all, so this is why we have the dealer network and other independent software and hardware suppliers. They can spend time altering their own packages; obviously, when it might mean a sale for them! On the other hand, Commodore's Payroll is designed to be as universal as possible in it's application, and most probably would need no tailoring whatsoever.

Stock Control was the in-thing not too long ago, when it seemed that every new program that appeared was connected with stock control. Call me biased if you like, but I truly think that Commodore's, written by Anagram Systems, is the best one on the market. Latest versions are written entirely in machine code for speed of operation, and the whole program revolves around a series of smaller programs called up from disk, so they've removed any restraints that might have been imposed by the 32K memory limit. Although there is a limit to what can be done with a stock control program, Anagram have gone about as far as it's possible to go. Other programs available include Trader, from the Bristol Software Factory (0272 314278), which is an integrated stock control, invoicing and sales accounts, and Comstock from Merchant Systems (01 353 1464). As with payroll programs, and indeed anything else that I mention it's best to ring them all up first for specific information and take it from

there.

Word processing has long been a battle between Wordpro from Professional Software (0707 42184), and Wordcraft from Dataview (0206 865835). Now a new, very low cost word processing package has entered the arena, namely Wordform from Landsoft, who were mentioned earlier in the paragraph on payroll programs. Not having seen the program in action I can't really comment on it, but £75.00 worth of machine code must be worth investigating. Apparently it was written by the proverbial 3 year old genius, and typed in blind. Ted Landsler, of Landsoft, told me "he just handed me the program, and said - this should work. Needless to say it did, perfectly!"

Moving smartly on, we come to data base management. Three packages lead in the field at the moment, namely Ozz, from Commodore, DMS from Compssoft (0483 39665) and Petaid from Stage One Computers (0202 295395). A large number of people ring up and ask for, say, an estate agents package, or some other kind of system that requires the retrieval and storage of large amounts of information. They then complain that a specially written package doesn't exist, forgetting that something like DMS can quite happily be set up to do what they want, linking, if necessary, to Wordpro or Wordcraft for standard letter output. So, if you are after a program that apparently doesn't exist, get your local dealer to give you a demonstration of one of the above data base packages tailored to your requirements - that's what they're there for.

What sort of packages can we expect to appear in the future? There appear to be two main programs looming up on the horizon. Silicon Office, from the Bristol Software Factory, which appears capable of doing anything bar playing space invaders, since it is a data base on the galactic scale, incorporating wordprocessing without the necessity to link to other programs as one of it's many outstanding features. This will only work on the new 8096, and will probably appear in its final form around the same time, i.e. October-ish. The other main contender will be The Manager, which will be distributed by Commodore themselves, and again will probably be out around October.

continued on Page 35.

Machine Code Programming

An Introduction to Basic, Machine Code, and Assembly Language: Part2. by Emily Berk

This time we will examine the 6502 hardware as it applies to the assembly language programmer, say a few words about the assembling process, do some numeric conversions, and then start to learn some assembly language. As the discussion becomes more technical, you may want to consult some references, such as 6502 Assembly Language Programming by Lance Leventhal (Osborne) or MOS Programming and Hardware Manual. I would be happy to answer questions, which can be routed to me via Peter Gerrard.

As you now know, an assembly language is intimately related to the microprocessor it controls. Since the whole purpose of the assembly language is to flip switches in the microprocessor, it is important that the programmer know which switches he is flipping and what it is those switches are doing. Figure 1 is a map of a 6502 microprocessor system as filtered through the eyes of an assembly language programmer.

Memory

Memory is an array of bits (switches) in which can be stored both instructions to the 6502 and data. The 6502 can read instructions and data from memory and it can write data into memory. The location of an instruction or a piece of data in memory is known as its address.

NOTE: Bit, short for binary digit, is the term used for any physical or logical entity that can exist in one of only two possible states at any instant. Programmers usually think of bits as being in state 0 or state 1, or, in other words, as equalling either 0 or 1. A group of 8 bits is known as a byte. The 6502 usually examines data in byte-sized chunks. 6502 assembly language instructions can be 1, 2, or 3 bytes long.

Registers

A register is a small set of associated bits that is contained inside the microprocessor. The 6502 has a number of 8-bit internal registers that can be used by the programmer. They are known as the X, Y, and A (accumulator) registers, and the PC and SP registers. (I will discuss the special uses of each of these registers as I go along.) Using registers instead of memory to store data is advantageous because the instructions used to access them are often shorter than other instructions and execute more quickly. Obviously, since there are only a few internal registers, and since they must often be used for special purposes, most programs will still need to use memory storage.

The ALU

The ALU, or Arithmetic Logic Unit, is the part of the 6502 which performs its arithmetic and Boolean calculations.

Condition Codes

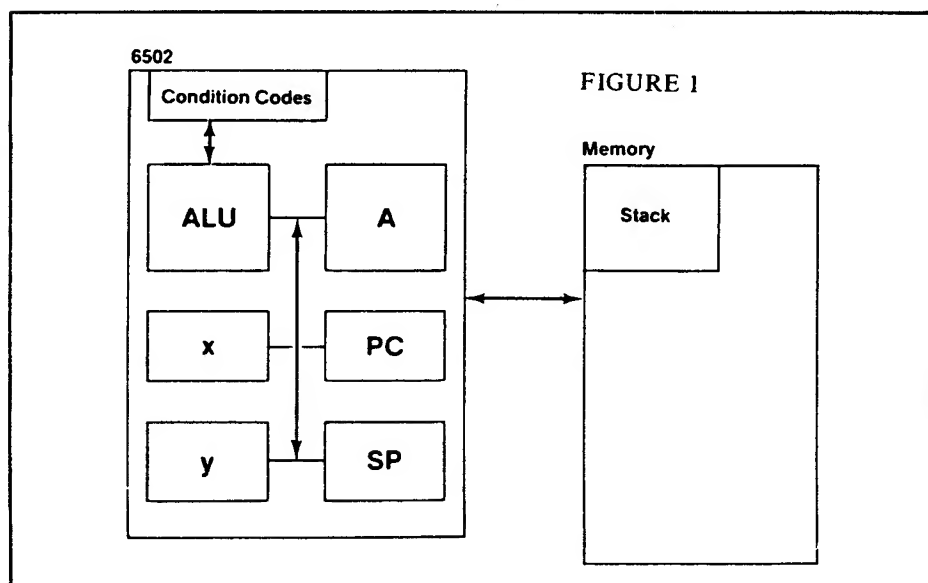
Operations in the ALU can have other affects. The 6502 Condition Code Register contains 4 bits. They are the

N bit, the C bit, the Z bit and the V bit. The N bit is set (i.e., equals one) if the result of an ALU operation was negative. The C bit is set if a carry resulted from an ALU operation. The Z bit is set if a zero resulted from an ALU operation. The V bit is set if an ALU operation overflows. Condition codes allow the program to make gotos (branches) conditional on the results of tests on data in the accumulator.

The Assembly Process

Before I go on, I will say a few words about assemblers. An assembler is a program (or person) which translates the mnemonic assembly language instructions into machine language that can be understood by the computer. Usually, this involves converting each mnemonic into numeric code - either hexadecimal, decimal or binary and storing the assembled machine language program at a known location. There are many machine language programmers who hand-assemble their programs and then enter them using either Basic POKE commands or the monitor feature of their microcomputers. Hand-assembling is a slow, tedious process. Care must be taken to do conversions carefully lest errors be induced. No syntax checking is done before programs are run, and if impossible operations are attempted, it may be as likely that the programmer has mistyped a digit as that his thinking was in error.

I, being lazy, much prefer to use a software assembler. An assembler can perform initial syntax and obvious error-checking that can prevent problems later on. It allows the programmer to think entirely in mnemonics and saves him hexadecimal headaches. An assembler also facilitates program documentation and alteration - especially usefully when one is first learning to use assembly language and when developing new programs. Choosing an assembler is a difficult task. The Commodore Software Encyclopedia lists two: The Assembler Development Package (p. 35) and EARL for the PET (p. 34). Check with your dealer for advice and a demonstration. Meanwhile, in my programming examples, I will try to supply both the assembly listing and the hexadecimal code. Deci-



mal conversions are left to the individual.

Numeric Conversions

By the way, do you know how to do numeric conversions? If not, it's time you learned. Let's start with binary. Binary, or base two, is a number system whose digits can be only one of two possibilities (that's why its binary, obviously). By strange coincidence, those possibilities are 0 and 1. Sometime in grade school most of us learned the concept of positional representation. That is, the number 8 in the rightmost digit of a decimal number means eight (8×10^0), but the number 8 in the place second to the right means eighty (8×10^1). In other words, each place in a decimal number is weighted (i.e., multiplied) by some power of ten. The same is true in binary, but instead of multiplication by powers of ten, we multiply by powers of two. In order to indicate that a number is in a particular number system, we follow it with a subscripted number equal to its base. So $10_{10} = 1 \times 10^1$, but $10_2 = 1 \times 2^1$. Translating from binary to decimal is fairly simple.

Multiply each binary digit by its positional weight, and then add your products. The multiplications don't get very complicated because you're always multiplying by either a one or a zero. So,

$101100_2 = 0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5 = 44_{10}$. Converting from decimal to binary is a bit more complicated. One way to do it is by iterated division. You divide the decimal number being converted by 2. The remainder (always a 1 or a 0) becomes the rightmost bit of the binary number. The quotient is then divided by two; this time the remainder is placed to the left of the previous remainder. Continue dividing until the quotient equals 0. For example, convert 45_{10} to base 2.

Binary numbers are fine for computers, but they do tend to get pretty long. For this reason, many programmers prefer to use hexadecimal, (known in the business as hex) base 16. Conversion from binary to hexadecimal is extremely simple, once you realize one thing. Remember that base two required two digits, 0 and 1, in which to represent all numbers. Decimal requires 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) to represent all numbers. Well, hex requires 16 digits

to represent all numbers. Unfortunately, the inventors of our number system (being decimal-type people) just didn't think to provide us with 16 unique one-place digits. (Remember, we can't use the 2-digit number 15 to represent a fifteen in base 16, because 15_{16} means $5 \times 16^0 + 1 \times 16^1$.) So in hexadecimal, we use the following digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Obviously, $F_{16} = 15_{10} = 1111_2$. Notice that all 16 of the hexadecimal digits can be represented in four binary digits. So, to convert from binary to hex, just group every 4 bits (starting from the right) and write down its hexadecimal equivalent. $44_{10} = 101100_2 = 00101100_2 = 2C_{16}$.

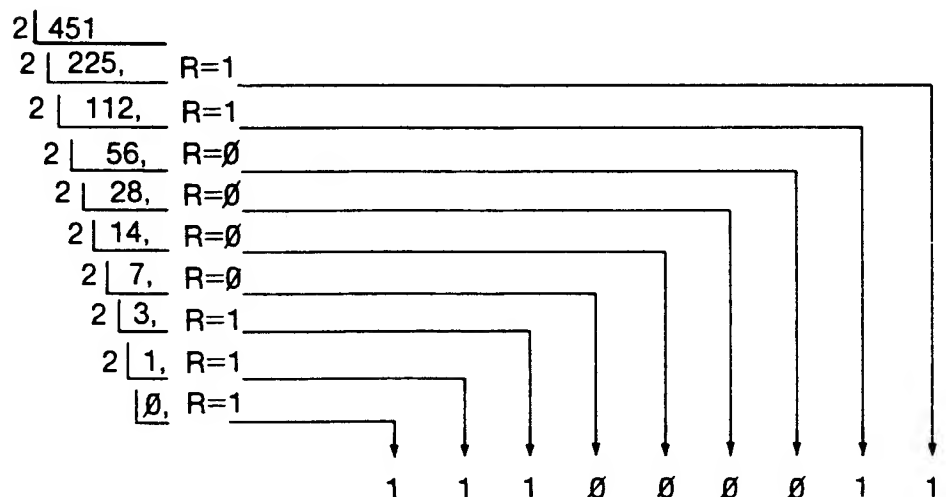
Converting back is just as simple. Convert each hex digit into 4 binary ones. There are other algorithms for all these conversions, some of them more suited for computer implementation. But, hopefully, you now understand the theory behind them. By the way, addition and subtraction are done in other number systems in the same way as they are done in base 10, except that you carry the two or the sixteen (by placing a one in the next column) instead of carrying the ten. So, to add $E_{16} + 3_{16}$, you add fourteen and three, equals seventeen. Seventeen is one sixteen (carry a one to the next place) and one = 11_{16} . The reason we have been doing all this number crunching is because assemblers list the locations of instructions and data in hexadecimal and because some of the instruction in the 6502 instruction set can only be understood when the data they affect is viewed as a set of bits. Next time we will present an assembler listing and analyse what its parts mean. But now, let's start to examine the types of instructions we have.

The 6502 Instruction Set

The instructions in the 6502 instructions set can be divided into the following categories:

1. Instructions that act on data in a single register or memory location.
2. Instructions that set or reset condition codes, and interrupt enables.
3. Instructions that transfer data from a register to a memory location, or from memory to a register, or between registers.
4. Instructions that change the ordinarily sequential flow of a program.
5. Instructions that compare the contents of a memory location with the contents of a register.
6. Instructions that operate on the stack.

An instruction can be divided into two parts - an op code and an operand. In general, the op code says what to do and the operand says to whom to do it. However, there are many 6502 instructions which do not require an explicit operand because the operand is implied in the op code itself. The format of an op code is invariant, while the format of the operand will vary because of the many different ways the location of an operand can be specified. The format of the operand part of an instruction is known as its addressing mode. Some op codes can be used with operands in a number of different addressing modes, while others require that their operands be in certain specific formats. The addressing mode of instructions that lack operands is called implied addressing. This type of instruction always affects a register, condition code, the stack or an interrupt. It never affects a memory location. All instructions in the implied addressing mode are 1 byte long. They include: INX, INY (increment the X or Y register), DEX, DEY (decrement the X or Y register), CLC (clear the carry bit), and TAX



and TAY (transfer the accumulator to the X or Y register).

Instructions in the immediate addressing mode are 2 bytes long. The first byte contains the op code. The second byte contains a decimal, binary, hexadecimal or octal (base 8) constant. The following op codes can be used with immediate addressing: ADC and SBC (add or subtract the constant in the next byte from the value in the accumulator, while taking into account the value of the carry), AND (logical-and the accumulator and the constant), CMP, CPX, CPY (compare the constant with, respectively, the accumulator, the X register or the Y register and set the condition codes according to the results you find), EOR (exclusive-or the constant and the accumulator), LDA, LDX, LDY (copy the constant into the accumulator, the X register or the Y register), and ORA (logical-or the accumulator and the constant). The results of all logical and arithmetic computations involving the accumulator are stored in the accumulator.

Don't worry if you have not memorized all these instructions. They are mentioned so you can begin to familiarize yourself with them. Next time we will examine some of them in more detail, and discuss some of the more complicated addressing modes and instructions available to you. ■

Machine Code Programs

Thanks go to Jan Owen for these – you may remember Jan for his article in last months educational supplement. They are all to assist in plotting, transferring blocks, drawing bars etc. on the PET screen. Simply type in the code, list the program to see how it operates, and first of all save a copy of it before running it. It's quite possible that you'll make a mistake typing the code in, so making a copy in case the program crashes is definitely a good idea. These are, regrettably, for Basic 2.0 only at the moment, but I'll try and publish the Basic 4.0 versions as soon as possible.

Some of you at the PET Show expressed a desire to know how to type this sort of program in – they'd seen many of them but were never able to use them simply because they didn't know how to. So, switch on the

PET, type SYS4 (RETURN), and you'll see a strange collection of letters and numbers. Don't panic! (As they say). Look at the four figure (or letter) number on the left hand side – in this instance it is always 033A. Type in M 033A 03AA (RETURN) and watch the screen fill with weird and wonderful things. Move the cursor back to where 033A is, and simply type in the values that appear in the listings in the appropriate positions on the screen. When you reach the end of line, type RETURN and move onto the next line. When you've filled up what is on the screen, and want to type in some more of the code, all you've got to do is look at the last line you've typed in (03AA in our example), move on about fifteen lines (e.g. to 0422) and type M 03AA 0422 (RETURN), and carry on typing in the values shown in the listing. When you finally get to the end, to save what you've typed in, you do the following:-

Look at the last four figure (or letter) number on the left hand side (e.g. 04CA in program four), and find out where the first AA appears in the listing. Count from 04CA to where that AA appears (counting is 0 to 9 and then A, B, C, D, E, F, and back to 0 again). In program four it appears at 04D1. The syntax for saving is:- S "0:PROG NAME", 08,033A,04D1

This assumes that you are saving the program onto drive 0 of device 8, and the program starts at 033A and finishes at 04D1. For saving programs to cassette, change 08 to 01 if you're using the first cassette deck, or 02 if you're using the second deck, and omit the drive number 0 within quotes. Obviously the beginning (033A) and end (04D1) values of the programs depend on the individual programs.

```
.. 033A A9 01 85 96 20 CC D6 E0
.. 0342 50 30 05 A2 35 20 57 C3
.. 034A 8A 4A 90 02 06 96 85 11
.. 0352 20 CC D6 86 4B A9 31 E5
.. 035A 4B 30 E6 4A 90 04 06 96
.. 0362 06 96 AA 18 BD 48 E7 65
.. 036A 11 85 11 B5 E0 09 80 69
.. 0372 00 85 12 B1 11 A2 10 DD
.. 037A AF 03 F0 04 CA D0 F8 E8
.. 0382 CA 8A 60 20 3A 03 05 96
.. 038A AA BD B0 03 91 11 60 20
.. 0392 3A 03 49 FF 05 96 49 FF
.. 039A 4C 8A 03 20 3A 03 45 96
.. 03A2 4C 8A 03 20 3A 03 25 96
.. 03AA A8 A9 00 4C 6D D2 20 7E
.. 03B2 7C E2 7B 61 FF EC 6C 7F
.. 03BA E1 FB 62 FC FE A0 00 00
.. 03C2 00 00 00 00 00 00 00 00
.. 03CA 00 00 00 00 00 00 00 00
.. 03D2 00 00 00 00 00 00 00 00
.. 03DA 00 00 00 00 00 00 00 00
```

```
.. 03E2 00 00 00 00 00 00 00 20
.. 03EA 3E F4 20 8D F6 A9 3A 85
.. 03F2 FB A9 03 85 FC 4C A4 F6
.. 03FA F7 E7 00 00 FF 00 00 24
.. 0402 04 01 00 8F 53 59 53 39
.. 040A 30 31 2C 58 2C 59 20 20
.. 0412 20 20 20 20 50 4C 4F 54
.. 041A 53 20 20 41 54 20 58 2C
.. 0422 59 00 47 04 02 00 8F 53
.. 042A 59 53 39 31 33 2C 58 2C
.. 0432 59 20 20 20 20 20 42
.. 043A 4C 41 4E 4B 53 20 41 54
.. 0442 20 58 2C 59 00 6A 04 03
.. 044A 00 8F 53 59 53 39 32 35
.. 0452 2C 58 2C 59 20 20 20 20
.. 045A 20 20 46 4C 49 50 53 20
.. 0462 20 41 54 20 58 2C 59 00
.. 046A 8D 04 04 00 8F 55 53 52
.. 0472 28 30 29 2C 58 2C 59 20
.. 047A 20 20 20 20 20 50 45 45
.. 0482 4B 53 20 20 41 54 20 58
.. 048A 2C 59 00 B0 04 05 00 8F
.. 0492 53 59 53 31 30 30 31 22
.. 049A 54 49 54 4C 45 22 20 20
.. 04A2 53 41 56 45 53 20 50 52
.. 04AA 4F 47 52 41 4D 00 D7 04
.. 04B2 06 00 8F 54 4F 20 45 4E
.. 04BA 41 42 4C 45 20 55 53 52
.. 04C2 20 20 20 50 4F 4B 45 31
.. 04CA 2C 31 36 35 3A 50 4F 4B
.. 04D2 45 32 2C 33 00 00 00 AA
```

```
.. 033A 20 CC D6 E0 28 30 05 A2
.. 0342 35 4C 57 C3 86 11 20 CC
.. 034A D6 86 96 38 A9 C7 E5 96
.. 0352 90 ED A8 29 07 AA 98 4A
.. 035A 4A 4A 60 20 F8 CD 20 C6
.. 0362 D6 86 96 A5 11 29 07 09
.. 036A 08 AA A0 03 66 12 66 11
.. 0372 88 D0 F9 A5 11 C9 28 30
.. 037A 05 A2 35 4C 57 C3 A9 19
.. 0382 E5 96 30 F5 60 A8 18 B9
.. 038A 48 E7 65 11 85 11 B9 E0
.. 0392 00 09 80 69 00 85 12 A0
.. 039A 00 BD BD 03 91 11 60 20
.. 03A2 3A 03 4C 87 03 20 5D 03
.. 03AA 4C 87 03 20 3A 03 A2 10
.. 03B2 4C 87 03 20 5D 03 A2 10
.. 03BA 4C 87 03 63 45 44 43 40
.. 03C2 46 52 64 65 54 47 42 5D
.. 03CA 48 59 67 20 00 00 00 00
.. 03D2 00 00 00 00 00 00 00 00
.. 03DA 00 00 00 00 00 00 00 00
.. 03E2 00 00 00 00 00 00 00 20
.. 03EA 3E F4 20 8D F6 A9 3A 85
.. 03F2 FB A9 03 85 FC 4C A4 F6
.. 03FA F7 E7 00 00 FF 00 00 28
.. 0402 04 01 00 8F 53 59 53 39
.. 040A 32 39 2C 58 2C 59 20 20
.. 0412 20 20 20 20 50 4C 4F 54
.. 041A 53 20 20 22 C0 22 20 41
.. 0422 54 20 58 2C 59 00 4F 04
.. 042A 02 00 8F 53 59 53 39 33
.. 0432 35 2C 58 2C 59 20 20 20
.. 043A 20 20 20 50 4C 4F 54 53
.. 0442 20 20 22 DD 22 20 41 54
.. 044A 20 58 2C 59 00 76 04 03
.. 0452 00 8F 53 59 53 39 34 31
.. 045A 2C 58 2C 59 20 20 20 20
.. 0462 20 20 42 4C 41 4E 4B 53
.. 046A 20 22 C0 22 20 41 54 20
.. 0472 58 2C 59 00 9D 04 04 00
.. 047A 8F 53 59 53 39 34 39 2C
.. 0482 58 2C 59 20 20 20 20 20
.. 048A 20 42 4C 41 4E 4B 53 20
.. 0492 22 DD 22 20 41 54 20 58
.. 049A 2C 59 00 C0 04 05 00 8F
.. 04A2 53 59 53 31 30 30 31 22
.. 04AA 54 49 54 4C 45 22 20 20
.. 04B2 53 41 56 45 53 20 50 52
.. 04BA 4F 47 52 41 4D 00 00 00
.. 04C2 AA AA AA AA AA AA AA AA
```



```

.. 033A 20 CC D6 E0 28 30 55 A2
.. 0342 35 4C 57 C3 86 11 20 CC
.. 034A D6 E0 C9 B0 F2 8A 4A 4A
.. 0352 4A 85 4B 8A 29 07 AA A9
.. 035A 80 85 12 A0 00 A9 18 85
.. 0362 4C A5 4C C5 4B 30 06 D0
.. 036A 07 BD 86 03 2C A9 A0 2C
.. 0372 A9 20 91 11 18 A5 11 69
.. 037A 28 85 11 90 02 E6 12 C6
.. 0382 4C 10 DE 60 20 64 6F 79
.. 038A 62 F8 F7 E3 20 F8 CD 20
.. 0392 C6 D6 86 96 A5 11 29 07
.. 039A AA A5 11 46 12 6A 4A 4A
.. 03A2 C9 28 10 9B 85 4B E6 4B
.. 03AA 38 A9 18 E5 96 30 90 A8
.. 03B2 B9 48 E7 85 11 B9 E0 00
.. 03BA 09 80 85 12 A0 00 A9 28
.. 03C2 85 4C C6 4B 30 06 D0 07
.. 03CA BD DB 03 2C A9 20 2C A9
.. 03D2 A0 91 11 C8 C6 4C D0 EA
.. 03DA 60 20 65 74 75 61 F6 EA
.. 03E2 E7 00 00 00 00 00 00 20
.. 03EA 3E F4 20 8D F6 A9 3A 85
.. 03F2 FB A9 03 85 FC 4C A4 F6
.. 03FA 38 2C 00 00 FF 00 00 3B
.. 0402 04 0A 00 8F 53 59 53 38
.. 040A 32 36 2C 58 2C 59 20 20
.. 0412 20 20 20 20 20 44 52 41
.. 041A 57 53 20 41 20 56 45 52
.. 0422 54 49 43 41 4C 20 42 41
.. 042A 52 20 58 20 41 4C 4F 4E
.. 0432 47 20 59 20 48 49 47 48
.. 043A 00 74 04 14 00 8F 53 59
.. 0442 53 39 31 30 2C 58 2C 59
.. 044A 20 20 20 20 20 20 44
.. 0452 52 41 57 53 20 41 20 48
.. 045A 4F 52 49 5A 4F 4E 54 41
.. 0462 4C 20 42 41 52 20 58 20
.. 046A 4C 4F 4E 47 20 59 20 55
.. 0472 50 00 98 04 1E 00 8F 53
.. 047A 59 53 31 30 30 31 22 54
.. 0482 49 54 4C 45 22 20 20 20
.. 048A 53 41 56 45 53 20 50 52
.. 0492 4F 47 52 41 4D 00 00 00
.. 049A 2C 59 00 C0 04 05 00 8F

```

Linking Machine Code Language & Basic

When programming the CBM computer the need often arises to incorporate some machine language code within a BASIC program. There are many ways this can be done, some simple, some quite complex. I will briefly try to explain here some of the alternatives available. Throughout this exercise we will use the following simple program to fill the screen with 'x' to demonstrate the methods we are using, (figure a, overleaf).

1) The machine program can be held simply in a list of data statements with each element of the statement corresponding to the decimal contents of a particular memory location. With this system it is usually necessary for you to hand assemble your machine language program and then convert the hand assembled code which will usually be in hexadecimal format to decimal values. These are then stored in DATA statements from which they are read and POKED into the correct memory locations. An example of the program listed above would be as shown in figure b, overleaf.

2) An alternative to the above method which requires less manual work is to store the machine code in hex format. In this way a simple algorithm is employed to convert from hexadecimal to decimal in the user program, figure c overleaf.

3) The disadvantage with both of the above methods is the need for data statements within the program and the fact that your machine code program will take up space addi-

tional to your normal requirements. In this case, for example, 26 bytes are used from 826 onwards. One way of combatting this with relatively short programs is to use the fact that the machine language program can be stored as part of the basic program in a REM statement. This technique involves typing into the REM statement the characters whose stored code is identical to the values required for our machine language program. Obviously character values that we cannot readily access from the keyboard (such as null, chr\$(0)) cannot be included in our program so the range of programs that can be entered in this way is minimal. Another limitation is that the program cannot be longer than 255 bytes (80 if entered from the keyboard) as this is the absolute maximum length of a program line.

To execute code entered in this way we place the REM statement as the first line of our program, irrespective of line number, and execute a SYS 1031 command.

4) Machine language segments once loaded can be saved along with a BASIC program by using the in-built Machine Language Monitor, or for those of you with BASIC 1.0 a copy of TIM. To do this simply find the locations of your machine language program and that of your BASIC program. Under normal circumstances the BASIC program will start at \$0400 (Hex) and end with a sequence of three "00" bytes. You can find these by issuing a 'm 0400-7fff' command and watching for the zero bytes. When you have found the extent of your BASIC and machine language programs simply save the area of memory that will encompass both programs. In the above example we would save from 826 (\$033A hex) to the end of our BASIC program say \$07f3 hex. We do this by issuing an 's' command and adding one to the ending address, as shown in figure d, overleaf.

The device can be any valid output device 01 = cassette, 08 = disk unit

When the specified file is loaded the machine language segment will also be included and no further loading will be required.

```

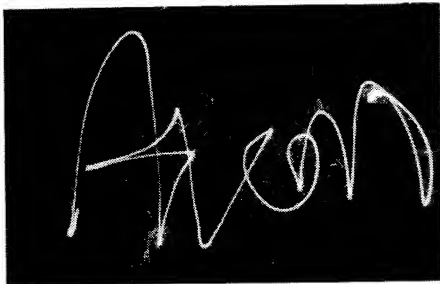
.. 033A 20 F8 CD 20 C6 D6 E0 28
.. 0342 30 05 A2 35 20 57 C3 86
.. 034A BB 20 CC D6 E0 19 10 F2
.. 0352 86 BC 18 BD 48 E7 65 BB
.. 035A 85 C0 BD E0 00 09 80 69
.. 0362 00 85 C1 20 CC D6 18 8A
.. 036A F0 D8 65 BB C9 29 10 D2
.. 0372 86 BB 20 CC D6 18 8A F0
.. 037A C9 65 BC C9 1A 10 C3 86
.. 0382 BC 60 20 3A 03 A0 00 B1
.. 038A 11 91 C0 C8 C4 BB D0 F7
.. 0392 18 A5 11 65 BB 85 11 90
.. 039A 02 E6 12 18 A5 C0 69 28
.. 03A2 85 C0 90 02 E6 C1 C6 BC
.. 03AA D0 DB 60 20 3A 03 A0 00
.. 03B2 B1 C0 91 11 C8 C4 BB D0
.. 03BA F7 18 A5 11 65 BB 85 11
.. 03C2 90 02 E6 12 18 A5 C0 69
.. 03CA 28 85 C0 90 02 E6 C1 C6
.. 03D2 BC D0 DB 60 00 D0 DB 60
.. 03DA 00 00 00 00 00 00 00 00
.. 03E2 00 00 00 00 00 00 00 20
.. 03EA 3E F4 20 8D F6 A9 3A 85
.. 03F2 FB A9 03 85 FC 4C A4 F6
.. 03FA F7 E7 00 00 FF 00 00 50
.. 0402 04 01 00 8F 53 59 53 39
.. 040A 30 30 2C 4D 2C 58 30 2C
.. 0412 59 30 2C 58 31 2C 59 31
.. 041A 20 20 20 54 52 41 4E 53
.. 0422 46 45 52 53 20 41 20 42
.. 042A 4C 4F 43 4B 20 46 52 4F
.. 0432 4D 20 4D 20 54 4F 20 58
.. 043A 30 2C 59 30 20 20 58 31
.. 0442 20 57 49 44 45 20 59 31
.. 044A 20 44 45 45 50 00 9F 04
.. 0452 02 00 8F 53 59 53 39 34
.. 045A 31 2C 4D 2C 58 30 2C 59

```

5) Especially on disk based systems another method that can be employed is to load the machine language section of code as a separate operation. This is accomplished by saving JUST the machine language portion of the code in the manner described above in example (4). The main BASIC program would then simply execute a LOAD or DLOAD command at some appropriate point in the execution of the program. NB: The main program is always restarted from the beginning if a LOAD or DLOAD is executed in program mode, however no variables are cleared. Consequently you will need to place a flag in the initial line of your program to test whether a load has already been performed. A sample first line could be something like this...

```
10 if fl=0 then x=1:
load"machinecode",8
```

This technique is very useful if the machine code segment resides at the top of memory and the user does not wish to use method 4 and save all the intervening unused memory area. Although this technique can be employed for cassette based systems the speed of the load operation may make it to slow by comparison with one of the other methods mentioned.



This is a photo of a sparkler. (Don't worry, it is nothing to do with machine code!)

(Figure a)

```
LDA #$00      ; set up start address in vacant zero page locns
STA $4B
LDA #$80
STA $4C
LDY #$00      ; reset Y register as pointer
LOOP1 LDA #'x  ; lda accumulator with 'x' character
LOOP2 STA ($4B),Y ; place 'x' on screen
INY
BNE LOOP2     ; if Y not back to zero go back
INC $4C       ; add 256 to base address by adding 1 to hi part
LDA $4C
CMP #$88      ; make sure we are not beyond the end of screen
BNE LOOP1     ; if not go back to do more
RTS           ; exit back to calling routine or BASIC
```

This code when expressed in hexadecimal is as follows ...

```
a9 00 85 4b a9 80 85 4c a0 00 a9 18 91 4b c8 d0 fb e6 4c a5 4c c9 88 d0 f1 60
```

(Figure b)

```
10 for i = 1 to 26
20 read a : poke 826+i,a
30 next
40 sys 826
50 goto 50 : rem pause when done until RUN/STOP pressed
9990 data 169,0,133,75,169,128,133,76,160,0,169,24,145,75,200
9995 data 208,251,230,76,165,76,201,136,208,241,96
```

(Figure c)

```
10 for i = 1 to 26
20 read a$: a=asc(a$): gosub 100: a2=a: a=asc(right$(a$,1)): gosub 100
30 poke 826+i,a2*16+a : rem poke with calculated hex value
40 next
50 sys 826
60 goto 60
100 if a<58 then a=a-48
120 if a>57 then a=a-55
130 return
9990 data a9,00,85,4b,a9,80,85,4c,a0,00,a9,18,91,4b,c8,d0,fb
9995 data e6,4c,a5,4c,c9,88,d0,f1,60
```

(Figure d)

```
s,'programname',dd,033a,07f4
↑
save      ↑      ↑      ↑
specified name  device to use  start address  ending address
```

Disk Use for Beginners

Hello it's me again! Thanks to some over zealous editing by the editor last time, all I have to do for this one is an introduction. Welcome to Part 5. There, that wasn't too bad was it.

SEQUENTIAL FILES - the basics

Sequential files are one method of holding data on disk ; other methods

will be covered in later articles.

A directory entry of a sequential file will look something like this:—

5 "DATA FILE 1" SEQ

The 5 is the number of blocks used by the file on the disk, and it can range from 1 to 670, 664, or 2052 (3040, 4040, 8050). The amount of information (number of characters) in the file can be worked out from that number.

For given number (N) the number of characters in the file (C) is $254 * (N-1)$
 $C \leq 254 * N$.

"DATA FILE 1" is the name of the file which can be any valid disk file name. SEQ tells you that the file is Sequential.

Creating a Sequential File

To create a sequential file the first

thing you must do is to OPEN the file, e.g. :—

```
OPEN 2,8,3,"0:DATA FILE 2,S,W"
```

The '2' is the "logical file number", or in other words the reference number, the '8' is the disk device number, and the '3' is the channel number. For sequential files use only 2 - 14, as 0, 1 and 15 have special meanings. The '0:' (or '1:') specifies the drive to be written to and must be present, otherwise the file will not be written correctly.

'DATA FILE 2' is the name of the file to be created. The ',S,W' also essential as this tells the disk unit the type of file required, 'S' being short for sequential, and that it is to be created or written, 'W' being short for WRITE.

It is a good idea to read the error channel after opening a file, to check that the file has opened correctly and that it does not already exist. I use a subroutine which reads the error channel and tells me if anything has gone wrong. (See the program listing at the end of the article). In an ideal world you should check the error channel after each PRINT #, however this can be very time consuming and so in many instances people do not take as much care as perhaps they should do. If your data is very valuable, or you are not overly concerned about time, then I suggest you check the error channel each time, especially if the disk is almost full.

To write the data into the file requires the use of the PRINT# command, e.g. PRINT#2, "DATA" will write "DATA" into file number 2 (note this is not complete yet). The above is correct for Basic 4.0, but Basic 2.0 requires some additional code. When all the data has been written the file MUST be closed using the CLOSE command, e.g. CLOSE 2., which will close file number 2. If the file is not closed you will not be able to read the information back from the disk. Again, check that there is no error.

Writing the Data

It is important that when data is read back, the program knows where to finish reading, and this can be achieved in several ways:-

1) Write a fixed amount of data so that there is always the same amount of data to be read back.

2) If a variable amount of data must be written, but the amount is known at the start, write the number of sets of data as the first entry in the file. When reading the data you would read this number and then read that many sets of data using a FOR ... NEXT loop.

3) Write a 'dummy' set of data at the end of the file. If the amount of data is not known until the end a dummy item can be written. This dummy item would be a value which could not be created by the program as ordinary data (999 is a popular number to use, or ZZZ if the data is alphabetic). When reading back you would check that this dummy value had not been reached, and when it has you know that you're at the end of the data on the disk.

4) ST. There is a reserved variable, ST, in the Basic operating system, which contains a value dependant on the last use of the IEEE 'bus. ST is set to 64 if the end of a file is reached when reading data. It is set at the point when the last item of data is read, NOT when you attempt to read some non-existent data.

Each method has its advantages and disadvantages depending on the application in mind. Personally I tend to use method 4 more than any other.

Simple Data Writing Routine

The array A() contains different numbers created by a different section of code (See program at the end).

```
3000 REM WRITE DATA TO DISK FILE
3010 OPEN 2,8,3,"0:DATA FILE 3,S,W"
3020 GOSUB8000: REM CHECK FOR DISK ERROR
3030 FOR B=0 TO 10
3040 PRINT#2,A(B);CHR$(13);
3050 GOSUB8000
3060 NEXT B
3070 CLOSE 2
3080 GOSUB8000
```

If you are using a 3000 series system, or have by any other method arrived at Basic 2, you will need to alter line 3040 (sorry...) to read :—

```
3040 PRINT#2,A(B);CHR$(13);
```

This is because in Basic 2 a PRINT# sends RETURN (CHR\$(13)) and LINE FEED (CHR\$(10)) characters. When data is read back the system only reads as far as the RETURN and so every item other than the first would be preceded by a LINE FEED character, which does not print on the screen (so everything

looks all right) but sorting routines would die a death.

What that additional code does is to suppress the RETURN and LINE FEED (using the ";"), adds its own RETURN (CHR\$(13)), and then suppresses the RETURN and LINE FEED again.

This routine uses method 1 to determine how much data there is. Next time I will show the other three methods in use.

Reading Data Back

As with writing you must open a file, e.g.

```
OPEN4,8,5,"DATA FILE 2,S,R"
```

Notice that the channel numbers need not be the same as when writing, and also the drive number need not be specified. Both drives will be searched if needed, starting with the drive last accessed.

To actually read the data you can use either INPUT# or GET#. As with the Basic GET, GET# only takes one character and INPUT# takes a series of characters. Generally you will probably be using INPUT#, e.g. INPUT#4,A,A\$ takes a number and a string from the disk, these items being separated by RETURN characters.

The routine to read the data corresponding with the write routine earlier is:—

```
5000 REM READ DATA FROM DISK FILE
5010 OPEN 4,8,5,"0:DATA FILE 3,S,R"
5020 GOSUB8000: REM CHECK FOR DISK ERROR
5030 FOR C=0 TO 10
5040 INPUT#4,D(C)
5050 NEXT C
5060 CLOSE 4
5070 GOSUB8000
```

No extra code is required for the Basic 2 reading routine.

Important

When using INPUT# to read information off the disk make sure that you do not have two RETURN characters with nothing between them (don't forget that second ";", with Basic 2!), and do not send null ("") strings when writing test data. If you do the computer will "hang" when you attempt to input the information. Note that CHR\$(0) is also a null string.

There must be a RETURN at least every 80 characters, otherwise the program will crash. With Basic 4, if the 81st character read from a disk file is not a RETURN the program will stop with a 'string too long error'. Basic 2 machines will crash with

a 'file not open error'. This is because the table which contains all the information about open files comes almost immediately after the input buffer. So, if more than 80 characters are INPUT the excess characters 'trample' over the file table. At the next attempt to access a file the file cannot be found because of the end of the previous string to have been input.

'Bye for now!

```

10 REM SEQUENTIAL FILE EXAMPLE
20 REM BY DAVID J. POOCK
100 REM A,B,C,D,N,M WORKING VARIABLES
110 REM EN,EM#,ET,ES DISK ERROR NUMBER,MESSAGE,TRACK AND SECTOR
120 REM A(),D() NUMERIC ARRAYS FOR DATA
1000 REM DEFINE VARIABLES AND OPEN ERROR CHANNEL
1010 A=0:B=0:C=0:D=0:N=0:M=0:EN=0:ET=0:ES=0:EM#=""
1020 DIM A(10),D(10)
1030 OPEN 15,8,15: REM OPEN ERROR/COMMAND CHANNEL

2000 REM CREATE DATA IN A()
2010 FOR D=0 TO 10
2020 A(D)=INT(50*RND(1))+1: REM RANDOM DATA (RANGE 1-50)
2030 NEXT

2100 REM PRINT ARRAY ON SCREEN
2110 PRINT"CLS"
2120 FOR D=0 TO 10
2130 PRINT A(D)
2140 NEXT
2150 PRINT"WRITING DATA"

3000 REM WRITE DATA TO DISK FILE
3010 OPEN 2,8,3,"0 DATA FILE 3,8,W"
3020 GOSUB8000: REM CHECK FOR DISK ERROR
3030 FOR B=0 TO 10
3040 PRINT#2,A(B),CHR$(13),
3050 GOSUB8000
3060 NEXT B
3070 CLOSE 2
3080 GOSUB8000

4000 REM PRINT CLEAR D() ARRAY
4010 PRINT"CHMS"
4020 FOR D=0 TO 10
4030 PRINT TAB(10),D(D)
4040 NEXT
4050 PRINT"(CRVS)READING DATA(RVF)"

5000 REM READ DATA FROM DISK FILE
5010 OPEN 4,8,5,"0 DATA FILE 3,8,R"
5020 GOSUB8000: REM CHECK FOR DISK ERROR
5030 FOR C=0 TO 10
5040 INPUT#4,D(C)
5050 NEXT C
5060 CLOSE 4
5070 GOSUB8000

5100 REM PRINT NEW D() ARRAY
5110 PRINT"CHMS"
5120 FOR D=0 TO 10
5130 PRINT TAB(20),D(D)
5140 NEXT
5150 PRINT"FINISHED"

6000 REM FINISHED
6010 CLOSE 15
6020 END
3000 REM ERROR CHECK ROUTINE
3010 INPUT#15,EN,EM#,ET,ES
3020 IF EN=0 THEN RETURN: REM NO ERROR CARRY ON
3030 PRINT"ICRDJICRDJCRVSJDISK ERROR",EN,EM#,ET,ES
3040 PRINT"ICRDJICRDJ ABANDONED"
3050 CLOSE2:CLOSE4:CLOSE15:END

```

MARK 2

```

2090 N=11 REM ELEMENTS 0-10

3000 REM WRITE MK2
3010 OPEN 2,8,3,"0 DATA FILE 4,8,W"
3020 GOSUB8000: REM CHECK FOR DISK ERROR
3025 PRINT#2,N
3030 FOR B=0 TO N-1
3040 PRINT#2,A(B),CHR$(13),
3050 GOSUB8000
3060 NEXT B
3070 CLOSE 2
3080 GOSUB8000

5000 REM READ MK2
5010 OPEN 4,8,5,"0 DATA FILE 4,8,R"
5020 GOSUB8000: REM CHECK FOR DISK ERROR
5025 INPUT#4,M
5030 FOR C=0 TO M-1
5040 INPUT#4,D(C)
5050 NEXT C
5060 CLOSE 4
5070 GOSUB8000

5100 REM PRINT NEW D() ARRAY
5120 FOR D=0 TO M-1

```

MARK 3

```

3000 REM WRITE MK3
3010 OPEN 2,8,3,"0 DATA FILE 5,8,W"
3020 GOSUB8000: REM CHECK FOR DISK ERROR
3030 FOR B=0 TO 10
3040 PRINT#2,A(B),CHR$(13),
3050 GOSUB8000
3060 NEXT B
3065 PRINT#2,9999 REM DUMMY VALUE
3070 CLOSE 2
3080 GOSUB8000

5000 REM READ MK3
5010 OPEN 4,8,5,"0 DATA FILE 5,8,R"
5020 GOSUB8000: REM CHECK FOR DISK ERROR
5030 C=0
5040 INPUT#4,M
5050 IF M=9999 THEN 5070
5060 D(C)=M:C=C+1:GOTO5040
5070 CLOSE 4
5080 GOSUB8000

5100 REM PRINT NEW D() ARRAY
5120 FOR D=0 TO C-1

```

MARK 4

```

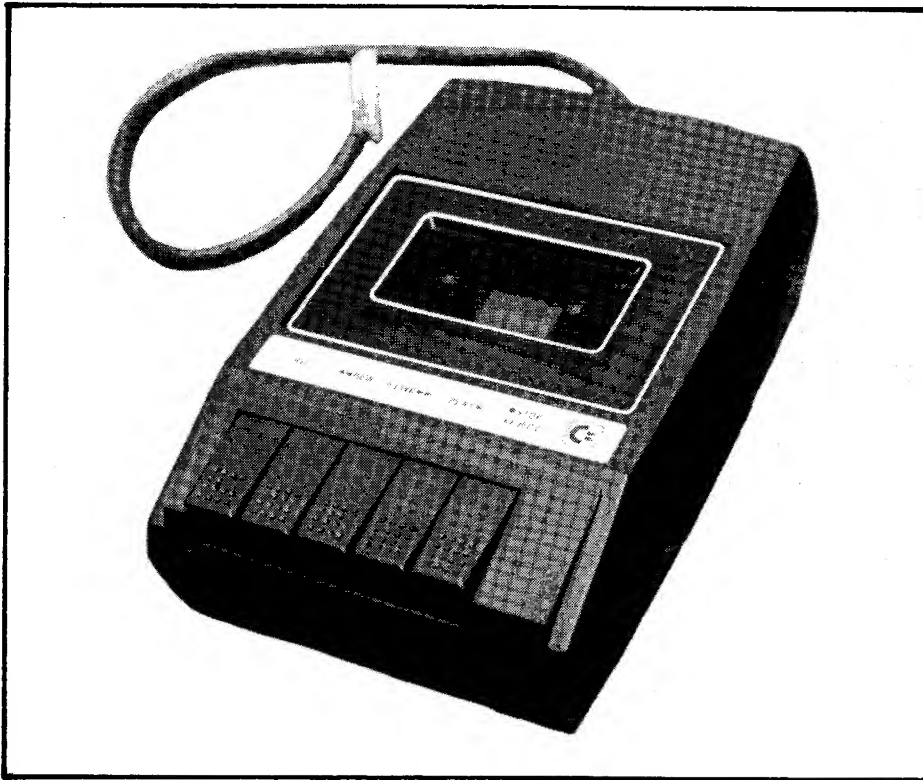
3000 REM WRITE MK4
3010 OPEN 2,8,3,"0 DATA FILE 6,8,W"
3020 GOSUB8000: REM CHECK FOR DISK ERROR
3030 FOR B=0 TO 10
3040 PRINT#2,A(B),CHR$(13),
3050 GOSUB8000
3060 NEXT B
3070 CLOSE 2
3080 GOSUB8000

5000 REM READ MK4
5010 OPEN 4,8,5,"0 DATA FILE 6,8,R"
5020 GOSUB8000: REM CHECK FOR DISK ERROR
5030 C=0
5040 INPUT#4,D(C):C=C+1
5050 IF ST=0 THEN 5040
5060 CLOSE 4
5070 GOSUB8000

5100 REM PRINT NEW D() ARRAY
5120 FOR D=0 TO C-1

```

Tape Programs



continued from Page 27

Having seen a demonstration of this in pre-production form the other day, I was very impressed by its capabilities. It will work on the standard 8032, and it is expected to produce a version for the 8096 before too long. Programs like this are very difficult to describe, since they don't fall into any set category, but they really create their own. I hope to be bringing you detailed reviews of both products shortly.

To end for the month, existing programs are currently being "upgraded" to work on the 8096. Things like Visicalc, Wordcraft, Wordpro (Wordpro 5 as it's called), DMS etc. will all be ready when the 8096 appears.

In the last two issues, our series of useful programs for a peripheral have concentrated on disk drives. So, for the benefit of those of you who are only using cassette decks, the following programs should be of use.

The first program will work on any version of BASIC, and is useful if you have lots of programs on tape, and want them automatically transferred to disk, rather than go through the process of loading them and saving them individually. Just load and run, and away it goes.

The second two are for testing cassette decks reliability, and as you can see are both in two parts. For each program, follow the steps outlined below.

- 1) Type in the BASIC program and save a copy of it.
- 2) Enter the monitor and type in the machine code part. Check very carefully and again make a copy in the normal way (i.e. just use SAVE).
- 3) The whole program is now ready to use, and can be loaded in future in one go as in step 2) you in fact saved the whole program, both BASIC and machine code parts.

Hope they're of use!

```

5 REM *****
10 REM *CASSETTE TO DISK DATA COPIER*
15 REM *
20 REM *      BY JAMES STRASMA      *
25 REM *      120 W. KING STREET    *
30 REM *      DECATUR, IL. 62521    *
35 REM *
40 REM *      AS OF MAY 5, 1980      *
45 REM *****
50 RS=CHR$(13):REM CARRIAGE RETURN
55 LF$=CHR$(10):REM LINE FEED
60 OPEN 15,8,15:REM COMMAND CHANNEL
65 GOSUB 230:REM ERROR CHECK
70 INPUT"DISK DRIVE # 0[3CL]";DR$
75 PRINT#15,"I"+DR$:REM INITIALIZE
80 GOSUB 230:REM ERROR CHECK
85 OPEN 1:REM READ CASSETTE
90 PRINT "[RVS]FOUND CASSETTE FILE
95 INPUT"DISK FILE NAME ?[3CL]";NM$
100 PRINT#15,"S"+DR$+"":NM$
105 GOSUB 230:REM ERROR CHECK
110 OPEN 2,8,2,DR$+"":NM$+"S,W"
115 GOSUB 230:REM ERROR CHECK
120 GET#1,DA$:REM READ A BYTE
125 SA=ST:REM REMEMBER STATUS
130 IF SA=64 THEN 195:REM FILE END
135 IF SA=0 THEN 170:REM OK
140 PRINT "[RVS]STATUS ERROR"SA
145 STOP
150 REM LINE FEEDS FOUL UP THE DISK
155 REM TO DELETE LINE FEEDS,
160 REM REMOVE THE 'REM' AT THE
165 REM START OF THE NEXT LINE
170 REM IF DA$=LF$ THEN 250
175 PRINT#2,DA$;:REM WRITE A BYTE
180 GOSUB 230:REM ERROR CHECK
185 PRINT DA$:REM COPY TO SCREEN
190 GOTO 120
195 CLOSE 1:REM CASSETTE
200 CLOSE 2:REM DISK
205 GOSUB 230:REM ERROR CHECK
210 CLOSE 15:REM COMMAND CHANNEL
215 PRINT "[RVS]END OF FILE REACHED
220 END
225 RUN:REM CAN'T FALL THROUGH
230 INPUT#15,EN$,EM$,ET$,ES$
235 IF EN$<"02" THEN 255
240 PRINT"[RVS]DISK ERROR[OFF]";
245 PRINT EN$ "EM$ "ET$ "ES$
250 STOP
255 RETURN
READY.

```


TAPE WRITE - JIM BUTTERFIELD

```

100 PRINT"TAPE WRITE LEADER TAPE "
110 PRINT" # JIM BUTTERFIELD
120 PRINT" THIS PROGRAM WRITES A CASSETTE TAPE"
130 PRINT"WITH 'LEADER' SIGNAL."
140 PRINT"THE CASSETTE TAPE SO PRODUCED MAYBE"
150 PRINT"USED WITH 'TAPE TEST' TO EITHER:"
160 PRINT" --CERTIFY THE TAPE AS OK;
170 PRINT" --ALIGN TAPE HEADS OF OTHER CASSETTE"
180 PRINT" UNITS. IN THIS CASE, BE SURE"
190 PRINT" THAT YOU ARE WRITING ON A"
200 PRINT" PRECISELY ALIGNED TAPE UNIT."
210 SYS1472:END

```

```

.
.: 04E0 8E 00 00 00 24 58 80 05
.: 04E8 69 04 00 00 54 00 82 00
.: 04F0 00 00 00 00 24 24 24 24
.: 04F8 24 24 24 24 24 24 24 24
.: 0500 20 12 F8 78 A6 D4 CA F0
.: 0508 15 CE 13 E8 A9 90 8D 4E
.: 0510 E8 AD 40 E8 8E FA 00 29
.: 0518 EF 8D 40 E8 10 0B EE 11
.: 0520 E8 A9 34 8D 13 E8 8D F9
.: 0528 00 A9 6E 8D 90 00 A9 05
.: 0530 8D 91 00 58 20 F0 F8 2C
.: 0538 13 E8 10 F8 A2 02 A0 00
.: 0540 A9 20 95 B8 B5 B1 F0 06
.: 0548 94 B1 A9 A0 95 B8 CA 10
.: 0550 EF A5 B8 8D 7A 80 A5 B9
.: 0558 8D F2 80 AD 6A 81 06 BA
.: 0560 69 00 29 1F 8D 6A 81 20
.: 0568 29 F7 10 C7 30 C5 20 7A
.: 0570 05 2C 40 E8 2C 10 E8 4C
.: 0578 E4 E6 AE 49 E8 AD 48 E8
.: 0580 EC 49 E8 D0 F5 A0 FF 8C
.: 0588 48 E8 8C 49 E8 E0 FC 90
.: 0590 08 E0 FF D0 07 C9 50 90
.: 0598 0B E6 B3 60 E0 FE D0 10
.: 05A0 C9 60 90 0C A5 CC 29 FC
.: 05A8 F0 03 E6 B1 60 E6 CC 60
.: 05B0 A9 00 85 CC E6 B2 60 AA
.
.
READY.

```

Apologies Section

Many apologies to the people attempting to use Jim Butterfield's "Copy All" programme. My fault entirely : I forgot to include the machine code part of the program with the listing. In most cases this will not be required, but it is needed under some circumstances. Load the BASIC program, enter the monitor in the usual way, type in the code, and resave the entire program. Thanks to the person who originally rang me up to tell me of the error : your prompt action enabled this correction to go in this, and not October's issue.

TAPE TEST - JIM BUTTERFIELD

```

100 PRINT"TAPE TEST # JIM BUTTERFIELD"
110 POKE59468,12
120 PRINT:X$="LEADER":GOSUB500
130 X$="DATA":GOSUB500
140 X$="ERROR":GOSUB500
150 INPUT"TAPE UNIT":T
160 IFT>20RT<1GOTO150
170 POKE212,T
180 SYS(1280):END
500 PRINT"
510 PRINT"  " ← ";X$
520 PRINT"  "
530 RETURN
READY.

```

```

.
.: 05B4 00 00 00 00 00 00 00 00
.: 05BC 00 00 00 00 A9 01 85 D4
.: 05C4 20 47 F8 A9 70 8D C3 00
.: 05CC 78 A9 A0 8D 4E E8 A2 08
.: 05D4 20 9B FC A9 02 85 DE A9
.: 05DC 34 8D 13 E8 8D F9 00 8D
.: 05E4 49 E8 58 A9 70 8D C3 00
.: 05EC 20 35 F8 F0 F6 20 7B FC
.
.
READY.

```

?

```

.
.: 0CC0 00 00 00 00 00 20 CC FF
.: 0CC8 A2 03 20 C6 FF 20 70 0D
.: 0CD0 8E 40 02 A5 96 8D 41 02
.: 0CD8 20 CC FF A2 04 20 C9 FF
.: 0CE0 AE 40 02 20 5E 0D AD 41
.: 0CE8 02 F0 DA 85 96 4C CC FF
.: 0CF0 A9 00 85 5F 85 60 E6 5F
.: 0CF8 D0 02 E6 60 A2 0E 20 C9
.: 0D00 FF A2 50 20 5E 0D A2 03
.: 0D08 20 5E 0D A6 5F 20 5E 0D
.: 0D10 A6 60 20 5E 0D A2 01 20
.: 0D18 5E 0D 20 CC FF A2 0E 20
.: 0D20 C6 FF 20 70 0D 20 CC FF
.: 0D28 E0 30 F0 01 60 A2 03 20
.: 0D30 C6 FF A0 FF C8 20 70 0D
.: 0D38 A4 61 8A 99 7A 02 A6 96
.: 0D40 F0 F2 84 62 20 CC FF A2
.: 0D48 04 20 C9 FF A0 FF C8 B9
.: 0D50 7A 02 AA 20 5E 0D A4 61
.: 0D58 C4 62 F0 9A D0 F0 84 61
.: 0D60 A0 10 8A 20 D2 FF A5 96
.: 0D68 29 03 F0 03 88 D0 F3 60
.: 0D70 84 61 A0 10 20 E4 FF AA
.: 0D78 A5 96 29 03 F0 03 88 D0
.: 0D80 F3 60 44 47 00 00 00 00
.: 0D88 00 AA AA AA AA AA AA AA
.
.
READY.

```